Information Management software

IBM

Dynamic Warehousing: Data Mining Made Easy

- Mining information from your data warehouse environment
- Understanding the data mining opportunities
- Using DB2 Warehouse mining capabilities

Chuck Ballard
John Rollins
Jo Ramos
Andy Perkins
Richard Hale
Ansgar Dorneich
Edward Cas Milner
Janardhan Chodagam

Redbooks

ibm.com/redbooks

International Technical Support Organization

Dynamic Warehousing: Data Mining Made Easy

September 2007

SG24-7418-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (September 2007)

This edition applies to Version 9 of DB2 Warehouse, product number 5724-E34.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
Cube Views™
Database 2™
Distributed Relational Database Architecture™
DB2 Universal Database™
DB2®
DRDA®
Informix®
Intelligent Miner™
IBM®
IMS™
Red Brick™
Redbooks®
Redbooks (logo) ®
SOM®
WebSphere®

The following terms are trademarks of other companies:

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

DataStage, are trademarks or registered trademarks of Ascential Software Corporation in the United States, other countries, or both.

ALPHABLOX, and ALPHABLOX logo are registered trademarks of Alphablox Corporation in the United States, other countries, or both.

Andreas, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

EJB, Java, JDBC, JDK, JRE, JSP, JVM, J2EE, Lyceum, Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

ActiveX, Excel, Expression, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

In this IBM® Redbooks® publication, we discuss the methodology and selected techniques of embedded data mining and show how sophisticated technologies can be used in today's business environment to create significant business value. All this is enabled by the IBM DB2® Warehouse (DB2W) data mining capabilities. Using DB2W, we show examples of using data mining capabilities for such analytic functions as data modeling, scoring, and visualization. In addition, there are scenarios and examples that help in understanding where, when, and how to use data mining.

The objective of data mining is to dynamically extract valuable information from the data warehousing environment to enable more informed decision making. An objective of IBM and this IBM Redbooks publication is to enable the use of data mining by a wider range of users through education, improved data mining techniques, high function tools for preparing and analyzing data, and by embedding data mining capabilities directly into business applications.

Significant business benefits can be realized when key information is extracted from the vast amounts of business data that exists in the data infrastructure of any company. Historically, the focus has been on getting data into the business data warehouse infrastructure. But to get the value out of that data, the focus is now on getting the data out of the infrastructure and creating information for use in making more informed business decisions. Using DB2 Warehouse enables clients to perform data mining much easier and much more quickly, and makes data mining capabilities available to a much wider audience of users.

As examples, IBM data mining can help you detect fraud, segment your customers, and simplify market basket analysis. The in-database mining capabilities integrate with existing systems to provide scalable, high performing predictive analysis without moving the data into proprietary data mining platforms.

Capabilities explored are:

- Modeling. The data mining process starts with historical data being gathered and put through a series of mathematical functions to derive business rules such as: "Customers buying certain types of cheese are also likely to buy mixed nuts 18% of the time." The business rules are then collected together into a Model, which can have a few rule or tens of thousands of rules.

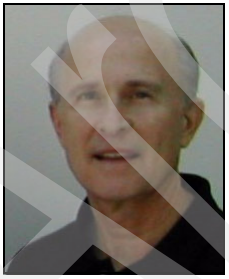
- Visualization. The business rules must be analyzed and verified by a business analyst to ensure they are accurate. IBM offers a unique Visualization tool to assist analysts in evaluating the business rules.
- Scoring. The verified business rules can then be applied to new data to determine the appropriate predicted outcome. For example, a new bank transaction enters the system and the fraud detection rules are applied against the data. The rules will predict the probability that the record is fraudulent. This process of applying the business rules is called Scoring. Scoring in real time allows businesses to catch fraudulent records faster, segment new customers and offer them better service, and detect defects quicker.

For years, data mining has been a valuable technology, but was considered complex and confined to a few very skilled analysts. With DB2 Warehouse, that has all changed. The significant value of data mining has been identified, the methodology defined, and the time to get started is now.

The team that wrote this IBM Redbooks publication

This book was produced by a team of business and technical specialists from around the world working at the International Technical Support Organization, San Jose Center.

The team members are depicted below, along with a short biographical sketch of each one:



Chuck Ballard is a Project Manager at the International Technical Support organization, in San Jose, California. He has over 35 years experience, holding positions in the areas of Product Engineering, Sales, Marketing, Technical Support, and Management. His expertise is in the areas of database, data management, data warehousing, business intelligence, and process re-engineering. He has written extensively on these subjects, taught classes, and presented at conferences and seminars worldwide. Chuck has both a Bachelors degree and a Masters degree in Industrial Engineering from Purdue University.

John Rollins, Ph.D., P.E., is the technical leader for data mining in the IBM Americas Software Group, where he has been involved in technical sales support, field enablement, intellectual capital development, and consulting for the past 10 years. Prior to joining IBM in 1996, he was an engineering consultant, professor, and researcher. He has authored many technical papers, a textbook, and six patent disclosures. John holds doctoral degrees in economics and engineering from Texas A&M University and is a registered professional engineer in Texas.

Jo Ramos is a Business Intelligence (BI) specialist for the BI Best Practices team in Dallas, Texas. He has eight years of experience with customer BI projects, providing support in consulting, architecture design, data modeling, and implementation of data warehouses and analytical solutions. Jo has worked with the majority of the IBM and Information Management products portfolio, specializing in analytical solutions. He holds a BS degree in Economics from the FURB University in Santa Catarina, Brazil.

Andy Perkins is a Business Intelligence (BI) specialist for the IBM TechWorks team in Dallas, Texas. He has 23 years of experience with customer database and BI projects, providing support in consulting, architecture design, data modeling, and implementation of data warehouses and analytical solutions. Over the years Andy has worked with the majority of the IBM and Information Management products portfolio on platforms ranging from the mainframe to the PC.

Richard Hale is a Worldwide Business Intelligence Executive for IBM specializing in analytics. He is involved with the development and global marketing and sales of IBM data mining products. He holds BS and MS Degrees in Economics from Auburn University. He completed the Lyceum™ of Executive Management at the Wharton School of the University of Pennsylvania, and is a Practitioner/Lecturer in Management Science at the University of Georgia. Richard has written a number of articles on econometric techniques and business issues addressed by advanced analytics for trade and academic journals, and has also authored a patent disclosure for IBM. Prior to coming to IBM he held sales and marketing positions for several high tech companies, and was a founding partner in one of the first data mining companies.

Ansgar Doerneich was lead developer for the data mining algorithms in IBM DB2 Data Warehouse Edition and IBM DB2 Warehouse. In July 2007, he joined Intelligement GmbH as a Senior Consultant for performance optimization and business intelligence. Dr. Dorneich has authored seven patent disclosures and many scientific and technical papers in the areas of quantum physics, high performance computing and data mining. Ansgar has 5 years experience with client data mining projects. He holds a doctoral degree in physics from Universität Würzburg and a Masters degree in economics from Fernuniversität Hagen.

Edward Cas Milner Ph.D., is a data mining and data analysis specialist in the Business Intelligence Best Practices group, which is in the IBM Americas Software Group. Prior to joining IBM, Cas worked in experimental particle physics, publishing over thirty papers. He has also worked as an investment quantitative analyst for mutual funds and hedge funds. His work at IBM has resulted in two patent filings and one patent award. Cas earned his B.S. and Ph.D. degrees in Physics from the University of Texas at Austin.

Janardhan Chodagam is a Business Intelligence (BI) Specialist in the Americas TechWorks Organization based in San Francisco, CA. He is responsible for enablement of DB2 Business Intelligence products, such as Alphablox, Cube Views™, and DWE in the Americas region. He has over six years of experience in application integration and data warehousing. Janardhan holds a Masters degree in Petroleum Engineering from the University of Texas at Austin.

Other Contributors:

Thanks to the following people for their contributions to this project:

From IBM Locations Worldwide

- Peter Bendel, Data Mining Architecture and Development, Boeblingen, Germany.
- Toni Bollinger, Data Mining Development, Boeblingen, Germany.
- Benjamin Leonhardi, IT Developer, Business Intelligence and Data Mining, Boeblingen, Germany.
- Pat Moffatt - Program Manager, Education Planning and Development, Markham, ON Canada.

- Andreas® Wagener, Software Development, Business Intelligence, Boeblingen, Germany.
- Anissa Stephan, Data Mining Development, Boeblingen, Germany.

From the International Technical Support Organization, San Jose Center
Mary Comianos - Operations and Communications
Deanna Polm - Residency Administration
Emma Jacobs - Graphics
Wade Wallace - Editor

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbooks publication dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our IBM Redbooks to be as helpful as possible. Send us your comments about this or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review IBM Redbooks form found at:

  ibm.com/redbooks
- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

The business value of data mining

Data mining has evolved from the ethereal domain of the highly-skilled mathematician to the expert data miner's workbench tool and ultimately to widely accessible business applications. For decades, industry and academia have been engaged in far-reaching research and development of data mining. At the same time, businesses have been leveraging this research, exploiting a handful of algorithms that are most useful in finding information to help resolve business problems. Recent trends have made these algorithms and systems, which are rooted in solid research, available to a wide range of business users in easy-to-use forms. Large numbers of business analysts, who may not be data mining experts, can now solve high-value business problems using data mining technology embedded in database-resident business applications. In this book, we discuss the methodology and selected techniques of embedded data mining and show how sophisticated technologies can be used in today's business environment to create significant business value.

1.1 Origins of data mining

Data mining has its roots in computer science and cognitive behavior research. In the 1960s, computer scientists began researching what was then known as artificial intelligence but is now usually referred to as machine learning. They were investigating a fundamental question: Can computers be programmed to think? At the same time, cognitive psychologists were studying a similar question: "What is the nature of what we call thinking?" Each of these two lines of research often benefited from results obtained in the other. One of the many practical results arising from this synergism of research was the development of systematic data mining.

The concepts had already entered science fiction and popular culture. For example, the film *2001: A Space Odyssey*, in which a computer named HAL (derived, incidentally, from the letters preceding I, B, M) could think, see, hear, and speak, and vie with a human for control of a space ship, begged the question: "How would it be possible for a real computer to mimic human thinking?"

Computer science and cognitive research revealed the basic components, abilities and fundamental characteristics of thinking, among them:

1. Recognizing patterns
2. Classifying
3. Deciding
4. Predicting

For example, a fundamental problem in recognizing patterns is identifying clusters of dots such as the ones shown in Figure 1-1.

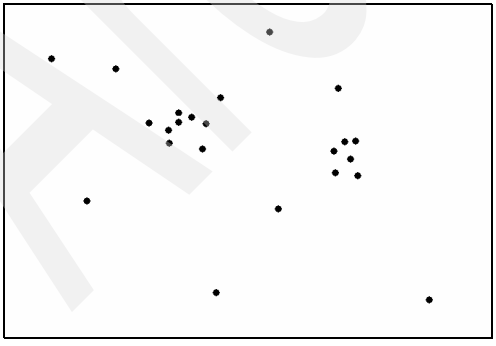


Figure 1-1 Pattern recognition problem

For humans, recognizing a pattern of clusters in this figure is trivial; even children can easily identify two clusters of points because this ability is innate to our species. Computers can perform the same task using precise algorithms to identify these clusters, but they can do so much faster, more accurately, and on vast quantities of data. The clustering technique can be highly useful in answering business questions, and for this reason two cluster-finding techniques are included in this book as data mining methods: demographic clustering and Kohonen neural clustering.

Note that the nature of the data in Figure 1-1 on page 2 has not been specified because the underlying computer pattern recognition technique is the same for data representing a variety of phenomena. For example, the data in the figure could be from such disparate sources as astronomical observations of stars, pixels of x-rays of potentially cancerous tissue, or a set of data representing customers' purchasing behavior. Analyzing any of these types of data may be a simpler task than outwitting a human to take control of a spacecraft, but it can still create a great deal of value.

1.2 Business value of data mining

The concept of clustering is relevant to behavioral psychology because human behavior often occurs in clusters. While all humans are unique at the most fine-grained level, at higher levels, for example, when engaging in a complex activity such as shopping, humans tend to display predictable and classifiable behavior falling into clusters according to behavioral and demographic characteristics. These characteristics include purchasing decisions, shopping frequency, age, gender, and income. Of course, the clusters are not rigidly delineated, but they are significant enough to provide business value because they are useful for predicting customer behavior.

The modern tools of data mining, connected with big data sets, arise from computer science research and behavioral psychology, but it is not necessary for a business analyst to know anything about these topics to produce valuable and actionable business results. Nevertheless, it is worthwhile knowing there is a long history of research establishing the validity of the concepts underlying data mining. For these reasons, data mining provides predictive power and sustained business value.

This business value is maximized when data analysis is driven by well-crafted business goals and questions. There are many patterns in business data, and they can certainly be found by data mining algorithms. But these patterns only have business value when they are directly related to a business question.

Data mining actively engages the database, making it an active participant in the analytical process. Using more traditional query techniques such as Online Analytical Processing (OLAP), a user can segment customers to be able to address their individual needs. The divisions may be based on queries by age, sex, geography, income, or other such characteristics, but the divisions are only as useful as the user's expertise in knowing which queries to pose and his ability to handle massive amounts of information with many dimensions will allow. Alternatively, using a data mining clustering technique implemented through a tool or application, the user can automatically discover distinct customer segments. There is no need to postulate what logical divisions may exist or how the segments should be structured. Data mining uses stored data and the intelligence provided by the clustering algorithm to actively assist the user in finding the most logical customer segments. No longer passively answering questions, the database becomes an active participant in deciding how the business's customers should be segmented.

When the database becomes an active partner in making decisions, users have to be confident in the information it provides. How is this confidence developed? In the early stages of a data mining project, it is often useful to "discover" information already known to be accurate. In this phase, data is validated, tool-learning occurs, confidence builds, and sometimes discoveries are made. For example, an apparel retailer knew that the climate in which a store is located was a primary driver of the kinds of merchandise that should be stocked. Not surprisingly, when they used clustering to segment their stores into logical groups, climate was the primary driver for most segments. But the greatest value came from discovery of a segment of stores where climate was relatively unimportant. Purchases of accessories, men's suits, and other such non-climate related merchandise by affluent shoppers made the segment unique and valuable. Business changes suggested by the analysis were to reformat the stores to lessen space allocated to outerwear, athletic shoes, and other lower-margin apparel merchandise to increase allocations of the merchandise more desired by the affluent shoppers.

In Chapter 3, “Case studies” on page 27, we present a set of case studies to illustrate a few of the many successful applications of data mining in solving business problems. The case studies demonstrate the use of a variety of data mining methods, including clustering, in various business scenarios and industries.

1.3 Data mining as a distinct form of data analysis

Data mining is a data analysis technique, but it is distinct from traditional statistical analysis, spreadsheet-driven analytics, or OLAP. The foundational analytical technique is query, or asking a question to receive an answer. The query may be repeated periodically to answer a recurring question as in a monthly report, or it may be a one-time (*ad hoc*) question to solve a unique business problem. Regardless of the kind of query used, the user is in a specific question-and-answer mode, which may be repetitive as answers generate new questions (drilldown). The user is the expert, and the database is a passive responder to questions, volunteering no unsolicited information.

OLAP in its many forms offers query users the ability to ask multidimensional questions to receive multidimensional answers. Instead of receiving simple answers to *ad hoc* questions, or two dimensional reports, users can exercise the many dimensions associated with information about which they have interest to find information to make better decisions. Just as in query, however, OLAP requires nothing more of the database than to be a passive responder to specific questions. The questions and answers are more complex, but the database remains a passive supporter of an expert user.

In data mining, automated pattern-finding algorithms organize the data into useful information, with less intervention by the user. In this way, the database role is enhanced. The discovery role of data mining means the organization can evolve to a higher level in using data, one with higher business value in which more people are empowered by information. This has been given the term “The Analytical Evolution”, and is illustrated conceptually in Figure 1-2.

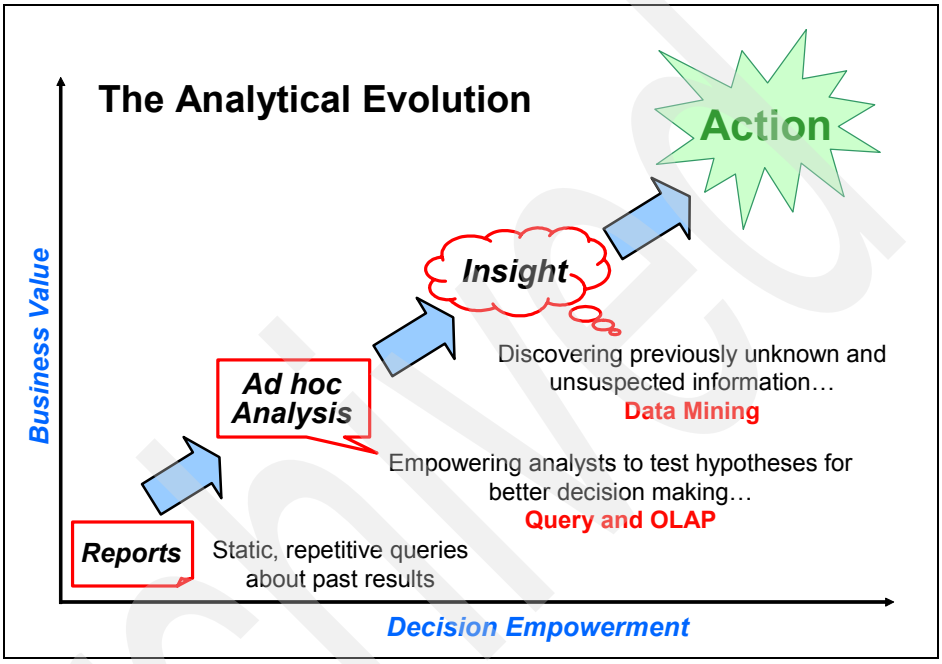


Figure 1-2 Analytical evolution

Data mining brings many new capabilities and functions to analysts. Some of the more important ones are:

- Discover previously unknown facts:
  - The tool learns and reports to the analyst.
  - The tool enables the data to reveal value.
- Organize and find value in large data stores for problems that are:
  - Too large to think about.
  - Too complex to understand.
- Alert in the present:
  - Tell the business about events that require attention.
  - Alert to predicted problems and not just those being experienced now.

- Predict the future:
  - Identify trends and anticipate problems and opportunities.
  - Competitive advantage with knowledge of the future.

1.4 Embedded data mining

Data mining used to be the province of highly-trained statisticians and mathematicians. Today, most businesses need to enable many business analysts to pose business questions and answer them using methodologies based in analytics. Embedded data mining is an approach to making sophisticated data mining methodology and technology available to large numbers of business users and enabling them to solve business problems or take better advantage of business opportunities.

Most businesses tend to have a spectrum of skills, as illustrated conceptually in Figure 1-3. Embedded data mining "democratizes" access to advanced analytics and enables users across the full spectrum to leverage the capabilities of the database to create, implement, and deploy data mining solutions to business problems. Within IBM DB2 Warehouse, this democratization is achieved by integrating all the necessary software components into a coherent and user-friendly platform, together with seamless interoperability with other software from other companies. As a result, solutions can be created, deployed, and managed faster and more efficiently, bringing greater value to the enterprise.

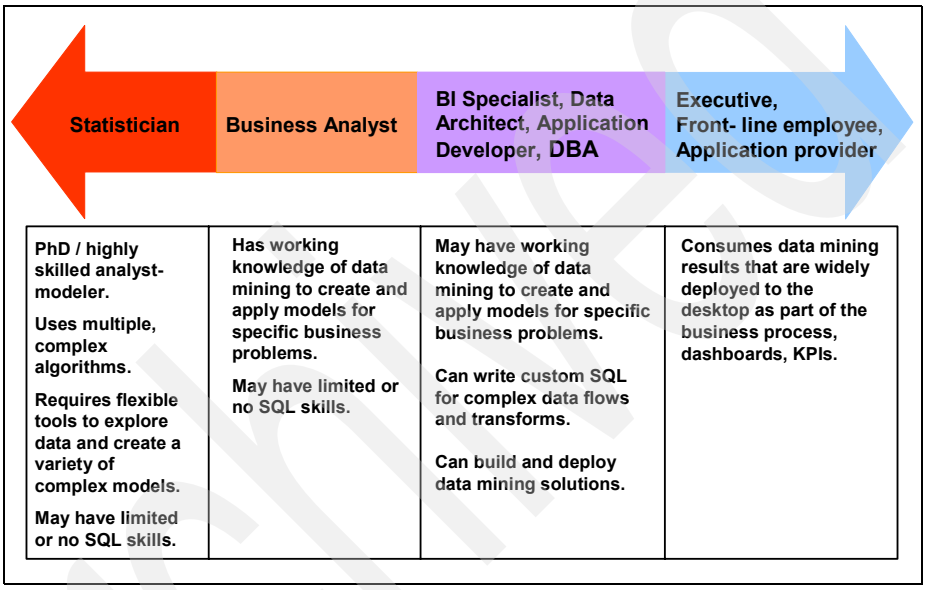


Figure 1-3 Spectrum of skills

1.5 In this book

In the following chapters of this book, we delve into the data mining process, the data mining methods and tooling in DB2 Warehouse, and data preparation for data mining. We show in detail how to construct data mining flows and interpret model results. We also discuss several case studies and a complete solution to demonstrate how data mining can be used to solve various types of business problems. Finally, we discuss emerging applications of data mining and provide a deep dive into the DB2 Warehouse mining algorithms.

2

Data mining as a process

In this chapter, we introduce the idea that data mining is not the workings of some magical and mysterious black box that is understood only by a handful of data mining wizards, but, in fact, it is a process that contains all of the typical elements of most any information technology project and that there is a huge class of business problems that can be solved using data mining methods that can be accomplished by mere IT mortals.

Here we look at a process that has been used successfully in many business situations. This process is introduced at a very high level, and then each major component of the process is discussed.

2.1 De-mystifying data mining

For many, perhaps most, people, data mining is a strange and almost mystical phenomenon. It is, therefore, a topic that is not well understood by most. This is because it can involve many very technical mathematical algorithms and require a detailed and complex analysis of the results.

Primarily for these reasons, data mining has not been as widely used as it could be. And so the value that can be gained from data mining has not been fully realized. But that is about to change. One way to start that change is to make the use of data mining much easier. This can be done by embedding much of the data mining processes in applications. Another is to de-mystify the process, making it more understandable by more people.

2.1.1 A historical look at data mining

This mystical perception may, in fact, have some historical basis. Data mining was born in academia by theoreticians with doctorates in subjects such as statistics developing the algorithms that are categorized as data mining algorithms today. In early data mining applications, these algorithms were custom-coded using programming languages such as FORTRAN. These early applications of data mining were necessarily very much focused on the implementation of the algorithms and remained very much academic in nature.

Over the years, various vendors developed new levels of tooling to make the application of these data mining algorithms easier to accomplish. This has resulted in a number of very good data mining workbench tools. However, the result of all of this is that data mining still tends to be a stand-alone activity done by a specialized staff, and data mining terminology is still very academically oriented. This all leads to data mining being viewed as very esoteric and something apart from normal IT projects.

A common perception of data mining, from a business use and IT staff perspective, is depicted in Figure 2-1. When there is a business problem that needs to be solved, the IT staff begins work on it. They understand the data warehouse, but once the business problem and data is handed off to the data mining team, it seems that they go away, perform some type of miracle, and emerge from the cloud with some type of analysis or recommendations that will enable some type of action. Then at some point in time, which may be sooner rather than later, the business problem may return based on changes in the business climate, and then the data mining wizards have to again be summoned to perform their miracle.

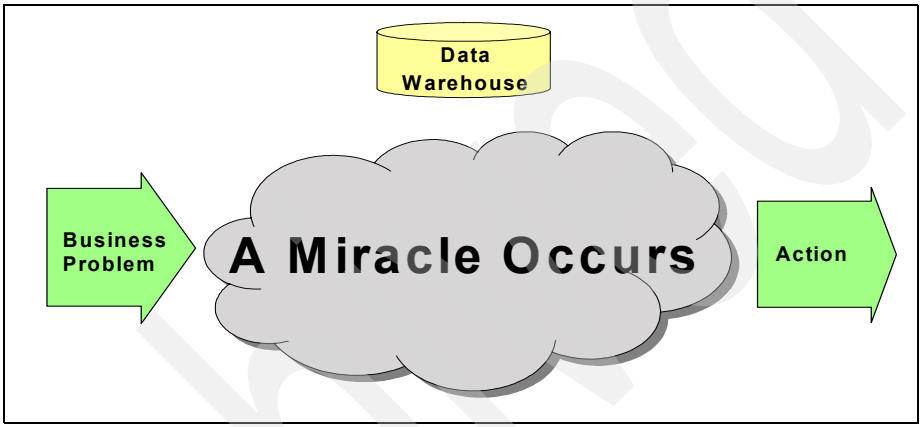


Figure 2-1 A common mis-perception of data mining

One of the major objectives of this book is to blow away some of the cloud and take a look at what is really happening underneath it. Take comfort in knowing that the vast majority of tasks involved in developing a solution to a business problem using data mining technology are basically the same as in any IT project.

2.1.2 Data mining as a process

The first thing to understand is that data mining is not the impenetrable dominion of the wise and mysterious data mining staff that it appears to be. In fact, it is very similar to any information technology project in that it is an iterative process, most of which involves typical IT types of tasks. Only a small portion actually contains the real data mining technology. And DB2 Warehouse nicely integrates that data mining technology into an integrated data warehousing toolset. For more on this topic, see Chapter 5, “DB2 Warehouse tooling for data mining” on page 51.

The overall data mining solution process can be segmented into three phases: defining the business problem, data mining, and taking action, as depicted in Figure 2-2. As with all IT projects, the first phase of the process is to identify and understand the business problem and define the initial general analytic approach. The second phase is the actual development work of preparing the data appropriate for the analytics, such as creating the mining model and deploying the mining model in a way most appropriate for the business problem and intended users. The third phase is again business oriented. Business actions will be taken and the results need to be monitored and measured against the original business problem.

For more information about the data mining process and to see a specific cross-industry data mining methodology, see the CRoss Industry Standard Process for Data Mining (CRISP-DM) listed in “Online resources” on page 520.

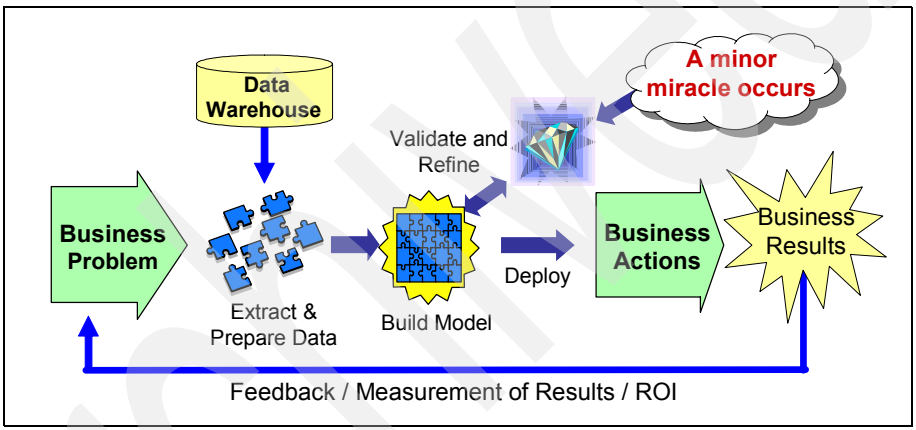


Figure 2-2 The high-level data mining solution process

2.2 Define the business problem

The data mining process begins not with data but with a problem to be solved. Since the purpose is to generate results (discoveries, insights, information, and models) that will help to solve the business problem, you must begin by clearly stating the business problem, translating it into one or more questions that data mining can address, and understand how the data mining results will be used in a business intelligence solution to improve the business.

This is the most critical part of the process. It is extremely important that this is a joint effort utilizing a strong team. Members of the team are the key players from the line of business (LOB) organization, key IT staff, a data analyst, and led by a strong project manager.

In Chapter 3, “Case studies” on page 27, we discuss the business problems resolved in a number of actual client case studies.

2.2.1 Identify the business problem

Data mining has many high-value uses, but it is not always the right tool for a particular purpose. For example, if the business need is to make the sales force more productive by reducing the time required to gather certain information, then the right solution might be a new reporting system based on real-time, multidimensional queries rather than data mining. On the other hand, if the need is to better understand customer behaviors and preferences to improve targeting for promotional campaigns, then the right solution might be based on customer segmentation and link analysis (associations and sequences). Understanding the business problem and how the data mining results will be deployed into the business enable us to determine the best approach for solving the problem.

Stating the business problem clearly and concisely forces a focus on the root issue of what needs to change to enable the accomplishment of a specific goal. Once you identify the root issue and determine that data mining is appropriate, then you can formulate one or more specific questions to address that issue. For example, to design a new targeted promotion, you might ask such questions as, What are the distinct profiles of behavioral and demographic attributes of our customers? Which of these customer segments looks best to target for the product that you want to promote? And for the target segment, what are the right customer attributes to use in designing an appealing promotion?

The key line of business members are extremely critical in this part of the process. Obviously, they understand their aspect of the business better than anyone else. Their input helps define the objectives of the solution as well as the success criteria in terms most appropriate for the business. They also help make definitions in terms of the business. For example, what is the definition of a business transaction for a department store that has cash registers in each department? In the eyes of the IT staff, a business transaction may be defined as one customer purchase at the cash register. However, the business user may define a business transaction as all of the purchases made across departments by a customer during one visit to the store.

2.2.2 Typical mining business problems

Data mining is used successfully in many different business and scientific fields, including retail, banking, insurance, telecommunications, manufacturing, health care, and pharmaceuticals. For example, in the retail, financial, and telecom industries, well-known uses of data mining include customer segmentation for targeted marketing and fraud detection, store profiling for category and inventory management, associations for cross-selling and upselling, and predictive techniques for customer retention and risk management. In manufacturing, data mining applications include profiling and predictive methods for quality assurance, warranty claims mitigation, and risk assessment. In health care and pharmaceutical research, high-value uses include associations, sequences, and predictive techniques for disease management, associations and prediction for cost management, and patient segmentation for clinical trials. In every case, the data mining results must be conveyed in such a way that someone can make a better decision or formulate an action to solve the business problem.

Here is a short list of typical questions that the business is asking that might lead to a data mining solution:

1. What do my customer look like?
2. Which customers should I target in a promotion?
3. Which products should I use for the promotion?
4. How should I lay out my new stores?
5. Which products should I replenish in anticipation of a promotion?
6. Which of my customers are most likely to churn?
7. How can I improve customer loyalty?
8. What is the most likely item that a customer will purchase next?
9. Who is most likely to have another heart attack?
10. What is the likelihood of a part failure?
11. When one part fails, what other part(s) are most likely to fail soon?
12. How can I identify high-potential prospects (lead generation)?
13. How can I detect potential fraud?

2.2.3 Identify the analytical approach

Understanding the business problem and how the data mining results will be used guides us in identifying the data mining approach and techniques that are most suitable to address the problem. Listening carefully to the questions that the business is asking will lead us to the appropriate analytical approach to identify the best mining method to use.

The types of business questions that are solvable by data mining techniques fall into two categories: discovery methods and predictive methods. If the business problem requires the we find useful patterns and relationships in the data, that will lead us to a discovery method. If the business problem requires us to predict some type of value, that, quite naturally, will lead us to a predictive method.

While every business problem appears unique and special, the vast majority can be solved by one of only five data mining methods. There are three discovery methods (clustering, associations, and sequences) and two predictive methods (classification and regression). In Chapter 4, “Data mining methods” on page 39, we discuss these methods in more detail.

Discovery methods

Discovery methods are data mining techniques that find patterns that exist in the data, but without any prior knowledge of what those patterns might be. That is, the objective is to discover the relationships that are inherent in the data. There are three discover methods: clustering, associations, and sequences.

- The clustering method groups data records into segments by how similar they are based on the attributes of interest. Clustering could be used, for example, to find distinct profiles of clients with similar behavioral and demographic attributes to create a client segmentation model.
- Associations is a type of link analysis that finds links or associations among the data records of single transactions. A very common use of the associations method is for market basket analysis. This is to find what items tend to be purchased together in a single market basket, such as chips and soda, for example.
- Another type of link analysis, sequential patterns, finds associations among data records, but across sequential transactions. A store could use sequential patterns to analyze purchases over time and, at checkout time, use that model to print customized discount coupons for the customers to use on their next visit.

Predictive methods

The predictive methods, classification and regression, are data mining techniques that can help predict some type of categorical or numeric value.

- Classification is used to predict values that fall into predefined buckets or categories. For example, they can predict whether a particular treatment will cure, harm, or have no effect on a particular patient.
- In contrast, regression is used to predict a numerical value on a continuous scale. For example, predicting how much each customer will spend in a year. If the range of values is 0 to 1, then this becomes a probability of an event happening, such as the likelihood of a customer leaving, or attriting, for example.

Combining methods

In some cases, a combination of data mining techniques is most appropriate. For example, if the business issue is to improve the effectiveness of a marketing campaign for an upcoming back-to-school promotion, then you may decide to perform customer segmentation using the clustering technique to identify a target group of customers, followed by a market basket analysis using the associations technique with the transactions only for the target group to find item affinities on which to base the promotion.

Mining and scoring

The five mining methods as discussed above are used against current data to create a data mining model. But what happens when new data comes in? The process of applying an existing mining model against new data is called *scoring*. Each of the five model creation methods has an associated scoring method used to apply against new data. Cluster scoring can be used, for example, to assign a brand new customer to the appropriate customer segment based on the existing cluster model.

Select the initial mining method(s)

As stated earlier, most business problems solvable by data mining techniques will use one or more of the five mining methods: clustering, associations, sequential patterns, classification, or regression. Understanding the business problem along with a high-level knowledge of the mining methods helps to define an initial approach to solving the business problem. We say initial because the approach may change due to other business or technical issues that may be discovered in later phases. Remember, this is an iterative process. As new information is discovered, it is not unusual to come back to revisit the business problem and make adjustments as necessary.

Using the short list of typical business questions from the previous example, the most common mining method, or combination of methods, to help answer the question can be noted as shown in the following list of examples:

1. What do my customers look like?
   Clustering
2. Which customers should I target in a promotion?
   Clustering
3. Which products should I use for the promotion?
   Associations or Sequential Patterns
4. How should I lay out my new stores?
   Clustering and Associations
5. Which products should I replenish in anticipation of a promotion?
   Associations
6. Which of my customers are most likely to churn?
   Classification or Regression
7. How can I improve customer loyalty?
   Clustering plus Associations
8. What is the most likely item that a customer will purchase next?
   Sequential Patterns
9. Who is most likely to have another heart attack?
   Classification or Regression
10. What is the likelihood of a part failure?
    Regression
11. When one part fails, what other part(s) are most likely to fail soon?
    Sequential Patterns
12. How can I identify high-potential prospects (lead generation)?
    Clustering
13. How can I detect potential fraud?
    Clustering plus Associations

2.3 Build and deploy data mining

The second phase of the data mining solution process accomplishes the actual data mining. This phase itself can be highly iterative. It consists of three major steps:

1. Data preparation
2. Creating and verifying the mining model
3. Deploying the model in some fashion

Figure 2-3 shows a more detailed view of the data mining phase.

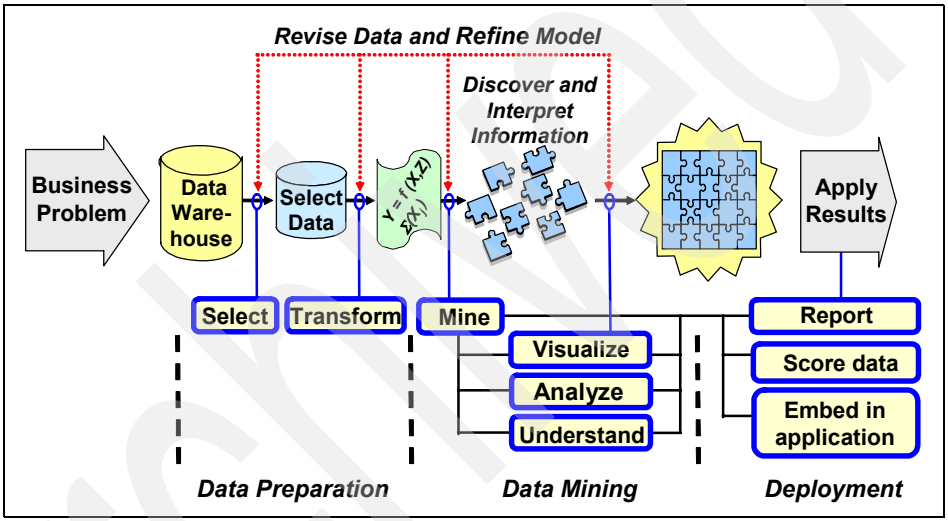


Figure 2-3 The data mining phase of the data mining solution process

Data preparation is exactly as it seems. It is finding and organizing the data for the chosen mining method. Once the data has been prepared, then the mining method can be invoked to create the mining model, which is then verified by the analyst. There will likely be one or more iterations to refine the data model. Once the model has been successfully verified, then it has to be deployed for use.

2.3.1 Data preparation

The data preparation step consists of identifying the data requirements, locating the data, and extracting and transforming the data into the appropriate format for the chosen mining method. This has historically taken the largest part of the overall time in the data mining solution process, which in some cases can approach 80% of the time. Proper data preparation is essential to a successful mining project and shortcuts should not be taken here. Spending time to correctly prepare the data will pay big dividends by reducing potential rework.

Identifying and locating the data

Understanding the data resources that are available to support data mining is integral to identifying the right analytical approach to solve a business problem. In most enterprises, vast amounts of data are collected and stored, typically in data warehouses and often in dedicated data marts, to support various applications. As a central repository of data resources, the data warehouse can greatly improve the efficiency of extracting and preparing data for data mining. Additional demographic data may be available from third-party or governmental sources to supplement the existing data and enrich the data mining analysis.

For any application, the data warehouse or specific data mart contains information (metadata) about the data, how it was derived, cleansed, and transformed, how it was formatted, and so on. The data and metadata together form a *data model* that supports the application and typically defines the data sources, types, content, description, usage, validity, and transformations.

Like any other application, data mining requires a data model to specify the data structure for each data mining technique. In most cases, the data model calls for a single *denormalized* table as the primary data table with perhaps one or more relational tables containing additional information, such as name mappings or hierarchies. The denormalized table may be either a physical table or a view of joined tables. The most common data types used in data mining are:

- Transactional data: Operational data generated each time some interaction occurs with the individual (*target*); typically contains a time stamp, target identifier, and item identifier (where an *item* may represent a product, event, or some other attribute).
- Behavioral data: Data about an individuals' behaviors or relationship characteristics, such as quantities purchased, amounts spent, balances, and number or rate of interactions.

- Demographic data: Information about an individuals' characteristics, such as age, gender, and location; sometimes obtained by third-party data providers.
- Production data: Data about production conditions, machines, and production chains, components, extra equipment, and information about delays, quality problems, warranty claims, and repairs.

The first task in data preparation is to identify what data elements are needed to accomplish the analysis using the chosen mining method. For example, if you are doing customer profiling, you will need basic information about the customer along with other relevant data appropriate to the problem, perhaps the total of their purchases over the last twelve months.

Once the required data elements are identified, then an inventory of those data elements needs to be generated. Do the data elements exist and, if so, where? If not, can they be obtained from an external source? The company data stewards maintain data models of the various data stores in the company and can help identify the location of various data elements. If the data elements exist in the data warehouse or, even better, a domain-specific data mart, that will help reduce the data preparation time. Even so, some amount of additional data preparation is required. Data elements that exist outside of the data warehouse will require more work and time to prepare.

If the data required for the chosen mining method simply does not exist and cannot be obtained, the chosen analytic approach may have to be revisited and modified.

Extracting and transforming the data

Some mining methods, such as clustering, classification, and regression, require a single record per individual (customer, patient, store, or event, as examples) with multiple columns containing the behavioral and demographic attributes of each individual. This format is referred to as the *demographic format*.

The associations method requires a transactions table with multiple records per transaction, where each record contains a transaction identifier and one of the items in that transaction. For analyzing sequences of transactions, a sequence group identifier is needed as well. This is referred to as the *transactional format*. Typically, a relational table containing a mapping of item numbers to their respective descriptions is also needed, as well as additional relational tables describing a hierarchy, such as item → group → department and the mappings of each group or department identifier to its respective description. The data model specifies how each of these tables must be structured to meet the needs of each data mining method.

In addition, predictive methods require two sets of historical data, including the outcome value, one for training the model and one for testing the model for accuracy in predicting the outcome value. Furthermore, some data elements may need to be derived or may require modification or redefinition, such as conversion of birth date to age. Other data structures may be needed to support name mappings or taxonomies (hierarchies) for certain methods.

Once the data elements have been prepared, the value distributions should be visually inspected to assess whether data distributions and ranges are plausible, and how to deal with them if they are not. Some elements may consist of a single value, a large number of missing values, or other inconsistencies. Data elements should be inspected individually (univariate) and in relation to one or more other data elements (bivariate and multivariate). This data quality assessment may require additional data preparation work.

2.3.2 Data mining

Up to this point, the process and associated tasks are pretty much the same as in any IT project and can be accomplished by a typical IT staff. Now comes the part where, to most IT folks, a little magic still exists. That is the creation and verification of the data mining model. A data analyst with an understanding data mining methods should perform this part of the process. The data analyst will need to understand how setting the various parameters affects the resulting model and how to interpret the data model to verify that it is correct.

After the data elements have been prepared, the data mining model is created by invoking the tool for the chosen mining method and supplying the appropriate parameters. Some mining methods may have multiple mathematical algorithms that may be used and may be selected using an input parameter. There are additional tasks for a predictive method. A predictive model requires a training step and a testing step.

The resulting model is created, stored, and will be viewed using the appropriate visualization tool that the analyst uses to validate the appropriateness of model to solve the business problem.

The visualizations present information about model quality, specific results such as associations rules or clusters, and other information about the data and results pertinent to the particular model. This information enables the data mining analyst to assess model quality and determine whether the model fulfills its business purpose. Improvements to the input data, model parameters, and modeling technique can then be made as needed to obtain a good model that meets the business objective.

For example, a clustering model intended to find high-potential customers to target for a new Family Plan cellular phone offer has one large cluster that contains 90% of all the customers with the remainder distributed across several very small clusters. Because it provides little distinctive information about the customers, the model is of poor quality and does not meet the objective of identifying high-potential customers with distinct behavioral and demographic profiles to support designing a promotional offer. Adjusting certain parameters and re-executing the mining run yields a model with clusters ranging in size from 30% to 1%, providing a set of distinct profiles each with enough customers to be meaningful in terms of the business objective.

As another example, a classification model intended to screen a hospital's patient database to identify those at high risk of developing a certain disease fits the training data extremely well and is highly predictive of the propensity to develop the disease. But one of the key predictors in the model is a costly test not covered by insurance except for patients already diagnosed with the disease. Thus, this variable is unknown for most patients in the database. Although this model is of high quality, it does not meet the objective of an early warning indicator for patients who have not developed the disease. Eliminating that predictor from the model yields a less accurate but more useful model that can be widely deployed to screen patients and identify those who would benefit from preventive treatment before they develop the disease.

As a third example, a regression model intended to predict the likelihood that a customer will default on a new bank loan fits the training data very well, is highly accurate with the testing data, and contains only those variables that are readily attainable at the time of loan application. But when the model predicts that a loan applicant has a high propensity to default, the loan officer is unable to state precisely why the loan application must be declined. Although this model is accurate and useful, the bank's legal department points out that the model does not meet the business need to be able to cite specific reasons for declining a loan. Building a decision tree classification model using the same data and variables yields a model of slightly lower accuracy but which has the advantage of explicit decision paths for predicting the likelihood that a loan applicant will default, thereby enabling the loan officer to meet the legal requirement of stating exactly why a loan application is declined.

Since modeling is an interactive and iterative process, model parameters may need to be adjusted to produce a better model, or additional data preparation may need to be performed.

In light of the overall project timeline, this activity is a relatively small, albeit important, task. See Chapter 7, "Data Mining in DB2 Warehouse" on page 139 for more discussion on how to create and score data mining models.

2.3.3 Deploying the solution

Businesses today are looking for ways to make data mining accessible to large numbers of line-of-business analysts and decision makers throughout the enterprise, or "bringing data mining to the masses." *Embedded data mining* is the concept of delivering data mining to business analysts through portals, reporting tools, and customized applications tailored to specific business areas, all within the database environment. Embedded data mining components implemented as database extenders for model building and applications facilitates the development and enterprise-wide deployment of data mining-based solutions by enabling data mining functions to be integrated into BI tools. Because data mining capabilities are implemented as database extenders, any reporting or query tool that can use SQL can become a data mining platform. Business analysts who are not accomplished statisticians can readily bring the power of data mining to bear on many different business problems.

The final step in data mining is to deploy the data mining results in the business as part of a business intelligence solution. Deployment is a extremely important because how and where the results are deployed is crucial to realizing the maximum value from data mining. Data mining results can be deployed in various ways to diverse business processes or systems, depending on the business needs:

- *Ad hoc* insight: Use data mining on an *ad hoc* basis to address a specific, nonrecurring question. For example, a pharmaceutical researcher may use data mining techniques to discover a relationship between gene counts and disease state for a cancer research project.
- Interactive insight: Incorporate data mining into a BI application for interactive analysis. For example, a business analyst may regularly use a targeted marketing application to segment the customer base, select a segment suitable for the next promotion, and perform market basket analysis specific to that segment. The item affinities and customer profile then feed analytical/reporting functions that identify the highest-value item relationships for the promotion and the best promotional channel given the characteristics of the target group.

- Scoring: Apply a data mining model to generate some sort of prediction for each record, depending on the type of model. For a clustering model, the score is the best-fit cluster for each individual. For an associations model, the score is the highest-affinity item, given other items. For a sequences model, the score is the most likely action to occur next. For a predictive model, the score is the predicted value or response.
  - Batch scoring: Embed a scoring function in a BI application for periodic scoring of a database using a data mining model and generating an action or alert when required. An application might be set up to periodically score a database of customers and proactively trigger an action in response to a change in an individual's score. For example, when an insurance customer's life situation changes, such as having a child, the application registers a change in that customer's scored propensity to purchase life insurance. An informational letter is automatically sent to the customer, and the agent is notified to follow up with an offer for a new or upgraded life insurance policy. Another approach is to integrate scoring into the Extract, Transform and Load (ETL) process to score new data as it enters the data warehouse.
  - Real-time scoring: Embed a scoring function in a BI application for real-time, on demand scoring of an individual during an interaction and immediately generating a recommended offer or action. For example, a call center customer service representative may call up a customer's record during a phone conversation, enter the current order, and receive a recommendation for a suitable item to offer the customer to complement the item just ordered.
  - Scoring in a business process: A business process is usually a series of business activities, some of which are automated and some of which may require human interaction. The BPM tool manages the flow of activities across organizational boundaries. Scoring may be integrated as a step within a BPM flow to apply the model to information as it flows through the process. For example, a business process that monitors for out-of-stock conditions that may require something more complex than a simple inventory threshold might have a predictive model that uses a number of variables, such as typical supplier lead times, weather, past sales history, and most recent sales, to predict that an out-of-stock condition might be imminent and trigger an alert to the store manager.

In Chapter 8, "Deploying a data mining solution" on page 213, we show various techniques for deploying a data mining solution.

2.4 Take action and measure results

In the first phase of the data mining solution process, the business problem was defined. As part of the business problem, some desired outcome or success criteria was also defined. Now that the data mining solution is deployed, the solution must be monitored and measured against the desired outcome to see if it has the desired change in business behavior.

Using the data mining solution, in whatever form it was deployed, gives some new insight to the users about their business. The user, based on the new insight, will either modify current actions or create new actions that cause a change in their business results. For example, a new promotion will be created based on the results of a market basket analysis application that embeds the clustering and associations methods.

The effect of these actions must be monitored over time and compared against the desired behavior. If the solution is not achieving the desired results, then it may need to be adjusted in some manner. So, there may be a need to iterate through the data mining solution process again to refine the business problem definition, the data requirements, the mining model, or the deployment vehicle. In actuality, this is the same for any IT development project.

Business condition changes may also dictate a change in the underlying business problem and might require another iteration of the data mining solution process.

Unfortunately, this monitoring and feedback loop is frequently left out of the process and, as the business changes, the data mining solution might become stale and greatly reduce the business value originally obtained. It is important to keep the data mining solution up-to-date to obtain the maximum business value.

In Chapter 9, “Solving a business problem with data mining” on page 237, we provide an end-to-end example of how to solve a business problem with data mining.

3

Case studies

In this chapter, we present several case studies to illustrate the data mining process for addressing various kinds of business problems or opportunities. The case studies are summarized in Figure 3-1.

INDUSTRY	USE CASE	APPROACH	DATA MINING TECHNIQUE
Retail	Promotion	Customer Segmentation and Market Basket Analysis	Clustering, Associations, Sequences
Retail	Store Formatting	Store Profiling	Clustering
Finance	Credit Card Customer Attrition	Prediction	Classification, Regression
Telecom	Promotion	Prediction	Classification
Healthcare	Cohort Selection	Cluster-Based Selection Methodology	Clustering
Retail	Store Performance & Merchandising	Store Profiling	Clustering
Insurance	Fraud Detection / Claims Analysis	Claim or Provider Profiling	Clustering
Healthcare	Disease Management	Prediction	Regression

Figure 3-1 Case studies

3.1 Retail

The following two case studies illustrate two common scenarios in the retail industry. The first case study deals with targeting selected customer segments for a promotion. The second deals with store profiling to guide store design in order to better serve their distinct clienteles and improve overall profitability.

3.1.1 Promotion targeting

A nationwide department store implemented an analytical solution based on data mining and in-line analytics. The overall goal was to increase revenue by identifying distinct segments of high-profit customers (and others who behave like them) and targeting them with more personalized offers to enhance customer loyalty, increase interdepartmental shopping, and increase basket size and value. This scenario illustrates the analysis to support the planning of a promotion based on women's apparel.

Business requirement

The business requirement was to improve promotion effectiveness based on the women's apparel department. Specifically:

- Characterize distinct shopping behavioral segments for customers who have previously purchased women's apparel.
- For each of these customer segments, discover affinities among women's apparel and other items in other departments.
- Identify the next most likely purchase for each customer segment.
- Increase the average spend of customers in the most profitable clusters.
- Increase the average spend of the less profitable customers who look like the more profitable customers (that is, those in the same clusters) by cross-selling or up-selling them the products that the more profitable customers have bought.
- Increase utilization of the store card through promotions or partnering with bank cards.

Data mining approach

The data mining approach combined three mining methods. The first step in the data mining approach was to create a clustering model of women's apparel customers to discover distinct patterns of shopping behavior among these customers and identify one or more high-potential segments to target in a promotion. The second step was to perform associations analysis for each selected segment to discover item affinities specific to those customers. The third step was to perform sequences analysis for each selected segment to discover the next most likely purchases for the customers in each segment.

Data extraction and preparation

From transactions data, about 4 million customers who purchased at least one item in women's apparel during 2006 were identified. All transactions for these customers during this time period were extracted to generate a table containing about 80 million visits. A visit is defined as all of one customer's transactions occurring at a given store on a given day. For example, a particular customer may purchase some items in women's apparel, then go to the cosmetics department and make another purchase while still in the store. These two transactions, which represent two payments to different cashiers, constitute one visit.

A customer purchasing behavior table was created by aggregating visits by customer, yielding one record per customer. From the transactional data and customer information, many attributes were created to represent each customer's total spending, relative spending by department, shopping frequency, basket size, returns, VIP status (based on spending level and store card usage), and other metrics about shopping behavior.

Data mining analysis

Clustering was used to perform customer segmentation for the 4 million customers who purchased women's apparel. The customer segments characterized distinct shopping patterns of these customers across all departments.

Market basket analysis (associations and sequences) was performed for the three selected clusters to find items with strong affinities to women's apparel. Associations represent interdepartmental shopping patterns for a given visit, while a sequence represents the next most likely item to be purchased given one or more previously purchased items. These relationships form the basis for targeted offers (including real-time offers while a customer is in the store) and indicate which merchandise should be stocked in preparation for a promotion.

Data mining results

The VP of marketing identified five high-potential segments of customers. The selected segments represented 500,000 customers who shop frequently, may shop across multiple departments, and have high spending levels. The segments included many non-VIP customers whose shopping behavior was similar to VIP customers (those who have high spending levels and use the store card). The five segments are summarized in Figure 3-2.

Cluster Name	Average Annual Spending per Customer	Number of Customers in Cluster	Expected Number of Responders to Promotion @5% Response Rate	Expected Incremental Spending @5% per Responder	Expected Incremental Spending for Cluster
Active Men's & Women's Apparel Shopper	$ 1,100	326,533	16,327	$55	$897,967
Active Cosmetics & Holiday Shopper	$ 900	255,200	12,760	$45	$574,200
Active Women's Apparel and Cross-Store Shopper	$ 1,600	428,933	21,447	$80	$1,715,733
Active Women's Apparel with Store Card Shopper	$ 2,100	328,667	16,433	$105	$1,725,500
VIP & Active Women's Apparel Shopper	$ 3,200	257,067	12,853	$160	$2,056,533
Total		1,596,400	79,820		$6,969,933

Figure 3-2 Customer segments for retail promotion

The plan was to design targeted promotional offers to appeal to the distinct characteristics of each target segment as revealed by the clustering model. Targeted promotional offers were based on associations and sequences relationships. Assumptions were a 5 percent response rate to the offers and a spending increase of 5 percent of the average annual spending per customer in each segment. Given the average annual spending per customer in each cluster, the expected incremental spending of nearly $7 million in response to a promotion based on women's apparel was calculated as shown in Figure 3-2.

Business actions

The incremental spending amount was used by the marketing team to justify the promotion based on ROI. The promotion design included stocking requirements based on expected increases in sales of women's apparel and related items as indicated by the data mining results from associations and sequences.

3.1.2 Store profiling

A store profiling study was performed for a retail holding company in the UK. The holding company manages eight apparel retail brands, each with its own chain of stores. Each of the brands addresses a unique market segment, varying from young men's trendy fashions to traditional and large-size clothes for women. The largest of the eight brands has more than 550 stores and focuses on high-quality women's apparel for mid-market consumers. The stores carry a general line of merchandise, including dresses, blouses, pants, accessories, and maternity wear for young to middle-aged women. Their entry customer is an early- to mid-twenties woman who is moving from a trendy point of view to a more classic look. This brand is referred to in the UK as a "high street store," or one located downtown on the main street. The store profiling study focused on this brand.

Business requirement

The business requirement was to improve profitability of the brand while increasing utilization of the new data warehouse. Of four specific objectives (reduce markdowns, increase sales, reduce costs, and increase customer loyalty), the marketing and merchandising team identified the reduction of markdowns as the most urgent and decided to focus on it during this study.

Data mining approach

Excessive markdowns in a retail store are usually a symptom of an underlying problem of incorrect assortment, that is, the store is not stocking the merchandise that customers shopping at the store want to buy. To understand the distinct shopping behaviors at stores with chronically high markdowns, clustering was used to perform segmentation (store profiling) to discover one or more segments of high-markdown stores on which to focus marketing and merchandising actions.

Data extraction and preparation

Transactions for stores having high markdowns were extracted. The transactions were aggregated by store to create one record per store with attributes reflecting the shopping behaviors and demographics of each store’s respective customer base, for example, basket value, shopping activity by day of week, customer income range, and sales by merchandise type and margin. In addition, the stores were categorized as Oxford Street (high-margin merchandise sold to weekend and career women shoppers), City (mixture of selected high-margin merchandise and low- margin staple merchandise), Town (mixture of less high-margin and more low-margin merchandise than stocked by City stores), and Local (primarily low-margin stock with very little of high-margin accessories).

Data mining analysis

Clustering analysis revealed one particularly valuable cluster. This cluster included a number of Town stores that exhibited distinctly different shopping from typical Town stores. While most Town stores had peak shopping days on Thursday when their mostly blue-collar clientele did their shopping, the Town stores in this cluster had a second peak day on Sunday. The Sunday shoppers had a much higher-than-normal basket (transaction) value, with high sales of high-margin women's accessories. Furthermore, much of the lower-margin merchandise was sold only after being marked down.

Data mining results

Mapping these Town stores revealed the reason for the unusual shopping pattern. They were located in isolated areas not near a city, and they were also surrounded by smaller communities having only Local stores. Women in the small communities were coming to these Town stores on Sundays for a shopping holiday and were not focused on bargains or low-end merchandise. Thus, these Town stores were behaving like Oxford Street stores.

The VP of Marketing realized that the team had uncovered two problems. First, these Town stores were not serving their Sunday customers as well as they could with higher-end merchandise, so some sales were being missed. Second, these same customers, having been disappointed by the Town stores, would likely bypass the brand's Oxford Street stores when shopping in a city.

Business actions

The Town stores in the valuable cluster were reformatted like Oxford Street stores. This action alone generated sufficient incremental revenue in the first two years to pay for the data warehouse.

3.2 Finance

To improve customer retention, a credit card company wanted to develop a better way to identify valuable customers who are at high risk of attriting. In this case, attrition was defined as a customer's decision to close his account voluntarily, as opposed to involuntary closure due to default. By identifying high-risk customers, the company could target them for retention incentives before they attrit, thereby improving customer retention and reducing churn.

Business requirement

The business requirement was to create a predictive model capable of scoring customers to predict their likelihood of attrition. The model needed to be applicable to score the entire customer base periodically as well as to score individual customers in real time.

Data mining approach

Either classification or regression could be used to create a predictive model for attrition. Both modeling methods were used to see which method would yield a better model for this case. (Because the model would not be used for, say, deciding whether to extend or deny credit to customers, the company was not legally required to use a decision tree classification model, which generates explicit decision rules that can be easily explained. Thus, the determining factor for the model type was simply its accuracy.)

Data extraction and preparation

The customer database was queried to extract active customers as of 90 days prior to the current date. Each customer had up to 21 months of account history prior to this cutoff date. The three months since the cutoff date represented the "attrition outcome" period during which each customer either remained a customer or voluntarily closed his account (attrited).

For each customer, data included behavioral data on transactions and other account attributes (for example, balance, activity, credit limit, and delinquency) as well as demographic information about the customer (for example, age and income). Approximately one million transactional records were aggregated and combined with the demographic data, resulting in about 50,000 customer records, with one record per customer representing behavioral-demographic attributes. An "attrition outcome" variable (Yes, No) was created for each customer, based on whether or not he attrited during the 90-day attrition outcome period.

Because the customer base had a small proportion (2.1 percent) of attriters, an enriched training set containing 21 percent attriters was constructed to improve the model's ability to characterize the behavior of attriters. For model validation, a testing set containing the actual proportion of 2.1 percent was used.

Data mining analysis

Classification and regression models were created using the enriched training set and validated using the testing set. In the classification model (a decision tree), *cost matrix weighting* was used to increase the relative cost of mispredicting an actual attriter relative to mispredicting a non-attriter, thereby improving the accuracy of predicting the behavior of primary interest, that is, attrition.

Data mining results

The cost matrix weighting enabled the decision tree classification model to perform somewhat better than the regression model in identifying attriters, so it was selected as the attrition prediction model. The most important predictors for classifying a customer as an attriter or non-attriter were the current balance, length of time as a cardholder, cash limit on the card, customer income, and credit score, card utilization rate, credit limit, and customer age.

Business actions

The model can be used in a batch process to score the current customer base periodically, capturing any changes in their characteristics that may affect their propensity to attrit. The company can then make a retention offer or take action to reduce the risk of loss if the customer does attrit. Retention offers may be customized based on customer attributes.

The model also can be deployed in an embedded mining solution to score an individual customer in real time while interacting with the customer through a call center, Web site, or other touchpoint. The customer is scored based on his current activity, enabling the company to make a real-time retention offer if needed.

3.3 Healthcare

In this hypothetical scenario, a research hospital's goal was to reduce congestive heart failure (CHF) patients' likelihood of readmission within 90 days of hospitalization by implementing a new treatment program for high-risk patients. The first phase was to conduct a pilot program with a cohort of CHF patients suitable for the treatment. Given the available budget for the pilot program and the high unit cost of the treatment, clinical criteria were established for selecting a cohort of 500 patients. The clinical criteria, when used to query to hospital's patient database, yielded only 296 patients. Thus, the cohort needed to be expanded to include another 204 patients as similar as possible to the 296 selected by the clinical criteria.

Business requirement

The business requirement was to expand the cohort for the pilot program to 500 patients by identifying those 204 patients most similar to the 296 selected per the clinical criteria.

Data mining approach

A priori criteria were used to select a "desired group" of CHF patients, which needed to be expanded to include others who were as similar as possible to those in the desired group. Clustering-based Selection Methodology (CSM) was a suitable approach for this purpose.

- CSM is an adaptation of the Promotion Targeting Methodology developed originally for the retail industry. CSM entails building a clustering model for the desired group selected by *a priori* criteria and then applying the model to score all those not selected by the criteria. The scored individuals are ranked by how well they fit their respective assigned clusters, and enough top-ranked ones are selected to augment the cohort to attain the needed size. For more information, see Hale, C.R. and Rollins, J.B., "Process and Heuristic Statistic for Prospect Selection through Data Mining", SVL920030124US1 patent pending (Dec. 2003), and Kraemer, et al, "System and Method to Optimize Control Cohorts Using Clustering Algorithms", END920060045US1 patent pending (Oct. 2006).

Data extraction and preparation

The patient database was queried to extract the population of CHF patients, based on a set of diagnosis codes that define congestive heart failure. This query extracted 2,927 CHF patients. One record per patient was constructed from a combination of demographic information (such as patient age and gender) and transactional records containing details about such elements as each patient's prior hospitalizations, physician specialties, length of stay, and time since last admission.

Clinicians established criteria to select a group (cohort) of patients most suitable for the new treatment. A query based on the following criteria yielded a desired group of 296 patients:

- Visits during past year > 1 and
- Days since previous visit <= 60 and
- Age >= 55

Data mining analysis

The desired group contained 204 less than the design number of 500 for the pilot treatment program, so CSM was used to find the 204 patients most similar to those in the desired group. A clustering model was created using the 296 patients in the desired group.

The model was applied to score the remaining patients (2,631 of the 2,927 CHF patients in the database) and assign each one to his best-fit cluster. The scoring results included the best-fit assigned cluster ID and the confidence or “goodness of fit” to the assigned cluster for each patient. The scoring results were written to an output table so that the patients could be ranked by how well they fit their respective assigned clusters.

Data mining results

The top-ranked 204 patients in the output table were selected to supplement the desired group, yielding a cohort of 500 patients for the pilot treatment program. Because all patients in the desired group met the clinical criteria, every cluster in the model represented a profile of suitable patients for the cohort. Thus, the particular cluster to which a scored patient was assigned was not important. Rather, the important consideration in selecting the scored patients for the cohort was how well they fit into their respective assigned clusters, that is, how similar they were to those in the desired group.

The selection of the 204 additional patients was validated by comparing the rates of 90-day readmission of the CHF patients by group. If the clinical criteria were valid for selecting high-risk patients, then the readmission rate for the desired group should be higher than the overall rate for all CHF patients. If the CSM approach was valid for augmenting the cohort, then the readmission rate for the top 204 patients should be higher than the overall rate although less than the rate for the desired group. As shown in Figure 3-3, the relative readmission rates were as expected, validating the CSM approach in this case.

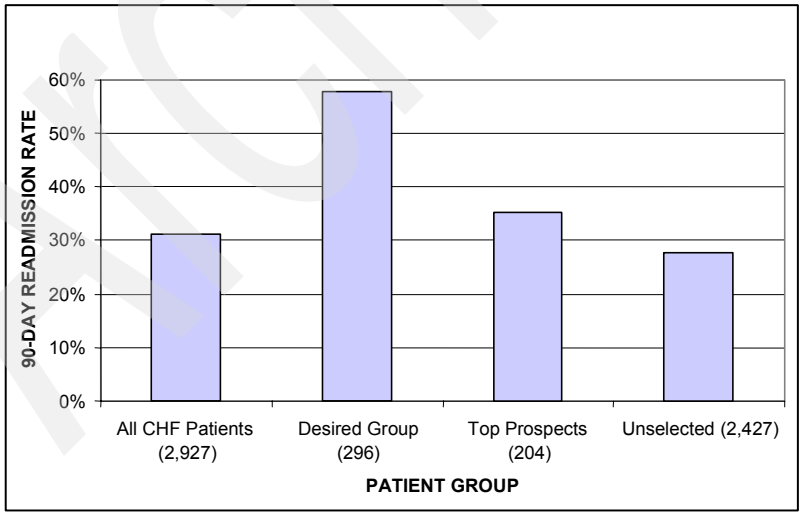


Figure 3-3 90-day readmission rates of CHF patient groups

Business actions

Unlike trial-and-error tweaks of selection criteria to expand a group, CSM provided a straightforward, data-driven method to select additional patients for the pilot treatment cohort. This methodology can be extended to similar situations where *a priori* criteria are used to select individuals for treatments or clinical trial cohorts.

It also can be used as the basis for reducing the size of a group if the criteria yield too many. A clustering model would be created using the desired group and applied to score the members of the desired group. These individuals would be ranked by how well they fit their respective assigned clusters, and the top-ranked ones would be selected to obtain the number needed for the cohort.

4

Data mining methods

Data mining methods fall into two groups called *discovery* and *predictive* techniques. This distinction derives from machine learning concepts concerned with how entities (such as computers and organisms) learn from sequences of inputs and outputs.

Discovery mining methods include *unsupervised learning* techniques. In unsupervised learning, an entity receives inputs (data) but has no target outputs other than the inputs themselves. Unsupervised learning can be viewed as finding non-random patterns among the inputs and constructing a representation (model) thereof that can be used in decision making. A classic example of unsupervised learning is clustering.

- Predictive mining methods include *supervised learning* techniques. In supervised learning, an entity receives a set of target outputs as well as the inputs, the goal being to construct a function (model) that produces the correct output given a set of inputs. This output could be a real number (regression) or a class label (classification). For further discussion, see "Unsupervised Learning" by Z. Ghahramani in O. Bousquet, G. Raetsch and U. von Luxburg, eds, *Advanced Lectures on Machine Learning*, Vol. 3176 of Lecture Notes in Artificial Intelligence, Springer Verlag, Heidelberg, Germany (found at http://www.gatsby.ucl.ac.uk/~zoubin/course05/ul.pdf and "Neural Networks and Statistical Models" by W. Sarle in Proceedings of the Nineteenth Annual SAS Users Group International Conference, April 1994.

In this chapter, we provide an overview of the data mining modeling methods included in DB2 Warehouse. For details on the algorithms underlying the mining methods discussed here, refer to Appendix A, “DB2 Warehouse data mining algorithms: a deep dive” on page 331. Data layouts appropriate for each method are discussed in Chapter 6, “Data preparation for data mining” on page 97, and implementation of the mining methods in DB2 Warehouse is discussed in Chapter 7, “Data Mining in DB2 Warehouse” on page 139.

4.1 Discovery methods

DB2 Warehouse offers three discovery mining methods: *clustering*, *associations*, and *sequences*. The latter two are sometimes referred to as *market basket analysis*, particularly in the retail industry.

4.1.1 Clustering

The clustering method seeks to find distinct clusters or groups of records having similar attributes. The records within a given cluster are as homogeneous as possible within the constraints of the parameters governing the execution of a particular clustering algorithm as well as the inherent capabilities of the algorithm itself, while the records in different clusters are as heterogeneous as possible. The records may represent customers, stores, warranty claims, or other individuals in the context of a business problem. For more information, “Techniques of Cluster Algorithms in Data Mining” by R. Grabmeier and A. Rudolph in Data Mining and Knowledge Discovery, Vol. 6, No. 4, 2002.

Clustering is an excellent technique to gain insights into behavioral and other attributes that characterize each profile represented by a cluster. In a given model, the most distinct clusters, and therefore the most interesting from a business perspective, tend to be the smaller ones. Behaviors represented by the large clusters tend to be known already, while the characteristics of the small clusters may have been obscured because they are relatively uncommon. Clustering also may reveal significant opportunities not considered in the original business question driving the clustering analysis.

A particularly interesting and useful result of clustering is the ability to find individuals who are similar to others in a given cluster but may differ in one or more key areas. For example, a bank interested in reducing churn (voluntary attrition) in their credit card customer base may use clustering to create a customer segmentation model. Using historical customer account information, this model includes a *variable* (a quantity or function that may assume any given value or set of values, generally represented by a column in a data table or file) indicating whether or not each customer has churned during a certain time period. The bank finds a cluster of profitable customers characterized by a much higher-than-average proportion of attriters, such as a high-churn cluster. This cluster, however, also includes some customers who have not yet closed their accounts. Because these customers are otherwise similar to those who tend to churn and therefore are at high risk, the bank can initiate actions to retain these customers before they churn. Furthermore, the bank can take into account the behavioral and demographic attributes of these customers to design tailored retention offers for different customers.

Once a clustering model has been constructed using historical data, the model can be applied to a new record to assign it to the best-fit cluster in a process called *scoring*. The scoring process may be used to score a single individual in real-time, for example, as the basis for making an offer while interacting with a customer. It also can be used to apply a model to many individuals periodically, for example, to identify cluster migration within a customer population as the basis for targeting certain business actions at those who migrate from one cluster to another. Another application of clustering is to identify individuals who are most similar to a desirable group, for example, expanding a prospect pool for a promotion or identifying the most suitable patients to expand the cohort base for a clinical trial.

DB2 Warehouse offers two clustering algorithms called *demographic clustering* and *neural (Kohonen) clustering*. Each algorithm has its advantages, and which one performs better in a certain situation depends largely on the data. Typically, a data mining analyst may try both algorithms to see which one yields a better model for a particular situation. See Appendix A, “DB2 Warehouse data mining algorithms: a deep dive” on page 331 for more information.

4.1.2 Associations

The associations technique finds links or associations among data records within a single event referred to as a transaction. For example, if the transaction represents a market basket, then the data records represent items purchased together in that basket, for example, chips and salsa. If the transaction represents an episode of illness for a medical patient, then the data records represent elements such as the symptoms, procedures, medications, and response to treatment.

An associations model generates combinations of items called item sets. Each item set contains one or more items and has a relative frequency of occurrence called *support* in the population of transactions being analyzed. These support values are used to construct *rules* that quantify the relationships or affinities among items occurring together in transactions. Including a *taxonomy* describing a hierarchy of items and higher-level groupings such as departments may also be useful in finding rules. In some cases, the analyst may be able to constrain which items are considered in the rules.

Associations rules are described using the following terms, which are illustrated in Example 4-1 for a rule [Orange juice] → [Cola]:

- Rule body: One or more items in a transaction, which imply the presence of another item; in this case, the rule body contains the first item (orange juice).
- Rule head: One item whose presence in the transaction is implied by the presence of the items in the rule body; in this case, the rule head contains the second item (cola).
- Support: Percentage of all transactions containing both the body item(s) and the head item.
- Confidence: Likelihood that the head item will be in the transaction, given the presence of the body item(s).
- Lift: Degree to which the confidence is greater (or less) than expected.

Example 4-1 Associations rule example

```
Transactions summary for a supermarket:
   Cola is purchased in 20% of all transactions.
   Cola is purchased in 60% of the transactions containing orange
juice.
   Furthermore, 3.7% of all purchases contain both cola and orange
juice.

Associations rule based on these conditions:
   [Orange juice] => [Cola] where orange juice is in the body and cola
is in the head

Properties of rule:
   Support = 3.7%
   Confidence = 60%
   Lift = 60% / 20% = 3

The lift of 3 means that cola is three times more likely to be
purchased when orange juice is also purchased, compared to cola
purchases across all
transactions.
```

An associations model can be applied in a scoring process to find the most likely item to be purchased along with one or more other items. This result is the basis for, say, a real-time offer for a particular item when some other item has been selected for purchase during an online shopping session.

Associations modeling is often done in conjunction with clustering. For example, customer segmentation may be performed to identify one or more customer groups to target in a promotion. Then associations analysis is performed using only the transactions for those customers for the time period relevant to the promotion (for example, the back-to-school shopping season), yielding a set of associations rules specific to the target customers for the right time period.

4.1.3 Sequences

The sequences technique finds links or associations among data records across multiple events (transactions) that are linked, for example, by customer. Each series of linked transactions is called a *sequence*. For example, a given customer may purchase a new house, then new furniture, and finally new landscaping.

A sequences model generates sets of sequences as well as item sets. Support values of these sets are used to construct rules that quantify the affinities among items occurring in sequential transactions. Concepts of support and confidence are analogous to those in associations but apply to transactions containing items rather than just to items. As with associations, a taxonomy may be used when constructing a sequences model.

Example 4-2 Sequences rule example

```
Transactions history for four customers (transaction groups) who
collectively purchased items A, B, C, D, E, F, and G:
   Customer 1: [B,E] => [C] => [G] => [A]
   Customer 2: [B] => [A,D] => [F]
   Customer 3: [F] => [D] => [B,C] => [A]
   Customer 4: [C] => [B,D] => [F,G]

Sequence rule of interest:
   [C] => [G] which indicates a purchase of item C followed by a
purchase of item G

Properties of rule:
   Support = number of transaction groups containing the sequence [C]
=> [G] divided by the total number of transaction groups
= 2/4 = 50%
   Confidence = number of transaction groups containing the sequence
[C] => [G] divided by the number of transaction groups containing the
body of the sequence
= 2/3 = 67%
```

Like associations, sequences modeling may be done in conjunction with clustering to discover the next most likely item(s) to occur, for example, the set of next most likely purchases for a target group of customers. A sequences model can be applied in a scoring process, for example, to score a customer in real-time to make an offer for the next most likely purchase.

4.2 Predictive methods

DB2 Warehouse offers two predictive mining methods. These are classification and regression, which are supervised learning methods.

4.2.1 Training and testing

During the model building process, supervised learning techniques employ mutually exclusive, distributionally similar data sets called *training* and *testing* sets. Since we want to use a predictive model to predict outcomes that have not happened yet, we are particularly interested in how accurate the model will be when applied to new data. Thus, we use the training set to build (train) the model and the testing set to validate (test) the model to assess its expected performance when applied to new records. Both the training and testing sets must consist of historical data, meaning that they both must contain the outcome (*target field* or *response variable*) to be able to construct and validate a predictive model.

The testing step is important because supervised learning methods may result in *overfitting*, meaning that the model fits the training data well but is not very accurate when applied to new data. Overfitting may occur if the training set happens to include spurious observations (data points) that do not represent population behavior, yet the training process fits the model to those observations. This situation is illustrated in Figure 4-1. In this example, the training set happens to include a 10-year-old individual with an unusually large shoe size. The model, represented by the solid red line, has been trained to the point where it fits this data point (red triangle) as well as the rest of the data points (green dots for everyone else). Although the overfitted model fits the training data very well, it will perform poorly for predicting shoe size as a function of age, at least for 10-year-olds. A better model would be the one represented by the dashed blue line. Although this model does not fit the training data as well as the overfitted model, it will perform better when applied to new individuals.

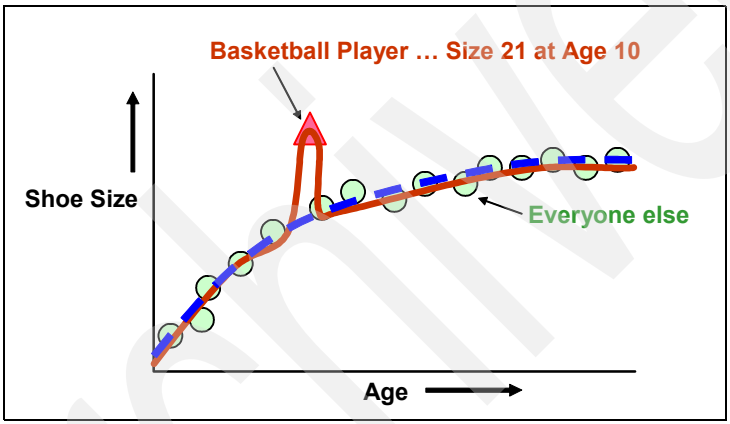


Figure 4-1 Example of overfitting during model training

4.2.2 Classification

Classification techniques assign or classify data records into categories of a class variable. The class variable is categorical and may have several possible values (for example, Yes/No or Help/Harm/NoEffect). An example of classification is customer churn (attrition) analysis to predict whether a customer will decide to stay or leave based on such characteristics as length of time as a customer, mix of products and services previously purchased, and demographic attributes. Another example is predicting which type of automotive part failure is most likely to occur, given attributes such as production processes, configurations, and history of previous problems.

One type of classification model, the decision tree, is illustrated in Figure 4-2. Each leaf of the tree, along with its respective branch, represents one decision path leading to an outcome. In this example, the model predicts that if customers have less than two years' tenure and use fewer than three services, then they are likely to leave (churn).

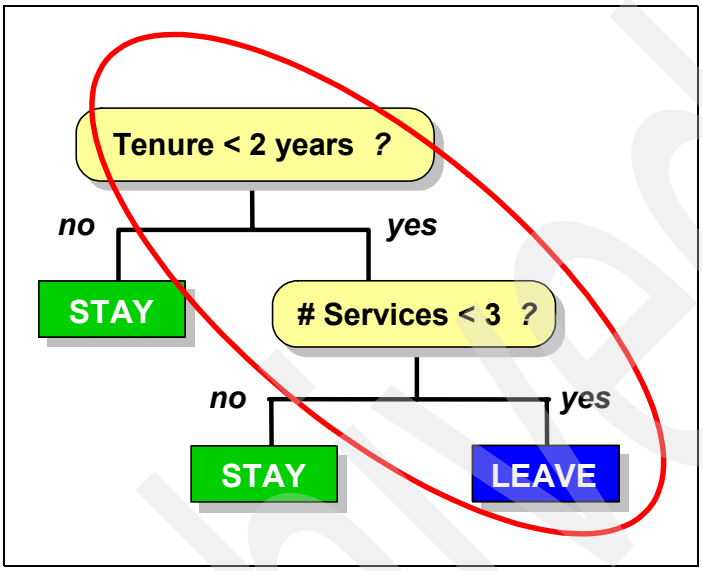


Figure 4-2 Decision tree example

A classification model can be applied to score one individual in real-time to predict a response or outcome. It also can be applied to score many individuals in batch mode, for example, to predict specific types of failures in warranty claims.

Classification techniques included in DB2 Warehouse are the decision tree, logistic regression, and Naïve Bayes. See Appendix A, "DB2 Warehouse data mining algorithms: a deep dive" on page 331 for details.

4.2.3 Regression

Regression techniques predict the value of a numeric dependent variable. The dependent variable may be a continuous value, such as annual revenue, or it may represent a binary response (0, 1), in which case the predicted value represents the probability (ranging from 0 to 1) of a positive response. An example is predicting a customer's annual expenditure based on attributes such as spending patterns, gender, and income. Another example is predicting the probability that a customer will respond to a promotional offer. Like other mining models, regression can be applied in a scoring process to help identify individuals to target for business actions.

Regression techniques included in DB2 Warehouse are Transform Regression, Linear Regression, and Polynomial Regression. See "Classification" on page 367 for details.

4.3 Choosing the right method

The business problem at hand dictates the kind of information needed to solve the problem and, therefore, which mining method or combination of methods is most appropriate. The five methods with their 11 algorithms available in DB2 Warehouse can be used to address most classes of business problems.

Clustering is a good method for gaining insights into behaviors and other characteristics of groups of individuals. It is particularly useful for discovering distinct combinations or profiles when the patterns that are most important in distinguishing individuals within the context of a business problem are unknown or poorly understood. Examples of business questions suitable for clustering analysis include:

- What do my customers look like, in terms of their spending patterns or demographic attributes?
- How do warranty claims differ across my manufacturing plants?
- Which current or prospective customers should I target in a promotion?
- Which healthcare providers may be submitting fraudulent claims?
- How should I lay out my stores across or within departments?

The associations method is useful for finding combinations of items, conditions, or other attributes that tend to occur together in a single event or transaction. It may include a response or outcome as one of the items in a transaction. Examples of business questions suitable for associations analysis include:

- Which product(s) should be used in a promotion to a target group of customers?
- Which products should be stocked together on the store shelves?
- Which transactions are likely to be fraudulent?

The sequences method is useful for finding combinations of items, conditions, and so forth, that tend to occur across transactions or events. It may include a response or outcome as the last item in a sequence. Examples of business questions suitable for sequences analysis include:

- Which product(s) should be used in a promotion to a target group of customers?
- Given a set of symptoms and prior treatments, what disease is a patient most likely to develop?
- Given the repair history, what part is most likely to fail next on a particular model of vehicle?

Classification is a good method when predicting a response or outcome represented by a categorical variable. It also provides explicit decision paths (classification rules), which may be desirable for legal or other reasons, such as justification for denying a loan to an applicant. Examples of business questions suitable for classification analysis include:

- Which customers are most likely to default on a loan?
- Which patients are most likely to have another heart attack?
- Which type of part failure is most likely to occur?

Regression is a suitable method when predicting a quantity or probability of an outcome represented by a numeric variable. Examples of business questions suitable for regression analysis include:

- What is the likelihood that a customer will churn?
- What is the expected amount of profit that a customer will generate?

A data mining analyst may use multiple methods together to solve a problem (for example, customer segmentation plus market basket analysis to identify target customers and promotional items). In other cases, different methods may be tried to see which one yields the best model to address a particular problem.

5

DB2 Warehouse tooling for data mining

In this chapter, we discuss DB2 Warehouse tooling to support data mining. Included is an overview of the DB2 Warehouse framework and functionality related to the process of developing a data mining solution.

The IBM Redbooks publication *Leveraging DB2 Data Warehouse Edition for Business Intelligence*, SG24-7274, provides a detailed discussion of the tooling for data warehousing and data mining, although it is relative to Version 9. Since that publication, the Data Warehouse Edition (DWE) has been renamed to DB2 Warehouse (DB2W), which is now at Version 9.5, and many of the components and features have been significantly updated.

It is for these reasons, as well as to provide a focus specifically on data mining, that this book has been written. So, let us take a look at the DB2 Warehouse framework and the tooling included for data mining.

5.1 DB2 Warehouse framework

DB2 Warehouse is an integrated information warehousing platform that includes the DB2 database, as well as tooling and capabilities for data extraction and transformation, data modeling, online analytical processing (OLAP), and in-line analytics and data mining. The primary components that comprise DB2 Warehouse are shown in Figure 5-1. The following list provides a brief description for those components:

- Data *Modeling*: Enables working with the physical metadata of the source and target databases, and the physical data model. The data modeling is done using the *DB2W Design Studio*.
- *Data Transform*: Enables development of SQL-based, intra-warehouse data movement, and transformations. The DB2W *SQL Warehousing* Tool (SQW) is used to develop the data transforms.
- *Data Mining*: Provides functionality to create, validate, visualize, and apply data mining models directly in the DB2 database.
- *OLAP* Enablement: Extends the data model to include metadata for multidimensional analysis (OLAP) and to enable optimization of query access to the database.
- *In-Line Analytics*: Enables creation of visual components and applications that utilize the analytical structures created by other components. The analytical components can be embedded into any Java™-based Web application or portal. The data visualization is developed using DB2 *Alphablox*.

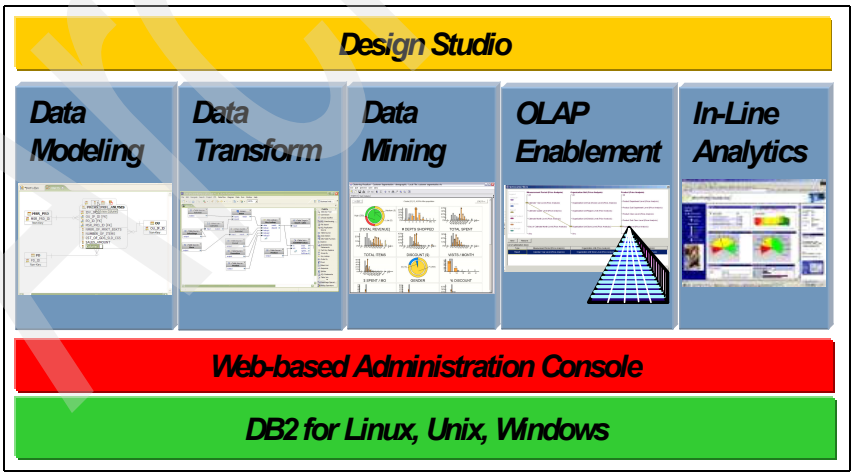


Figure 5-1 DB2 Warehouse: an integrated information warehousing platform

5.2 Design Studio

Design Studio is an Eclipse-based interface to DB2 Warehouse functionality. Design Studio serves as the client to provide access to all of the capabilities of the components described in Figure 5-1 on page 52.

5.2.1 Perspectives

Each window in the Design Studio contains one or more *perspectives* that can be selected. A perspective is a visual component of Design Studio and defines the views, editors, and displays of menus and toolbars specific to that perspective. When Design Studio is opened, the user is prompted to specify the desired workspace. The workspace is the directory containing all of the objects created by Design Studio. The user can create new workspaces simply by selecting **File** → **Switch Workspace** from the menu and entering a new path and workspace name. The user can also use this menu option to switch between workspaces. This capability facilitates organization and control over access to different workspaces by different users, such as teams working on different projects. Objects can be exported from one workspace and imported into another in Design Studio.

When Design Studio is started (for example, from the Start function in Windows®), the analyst can select the desired perspective from the menu bar, as illustrated in Figure 5-2. The default is the *Business Intelligence (BI)* perspective.

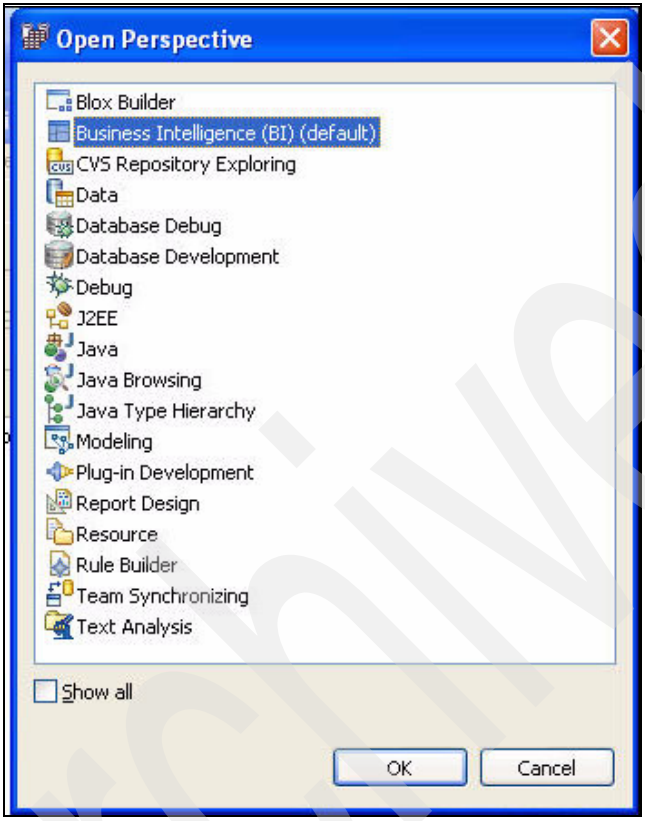


Figure 5-2 Selecting a perspective in Design Studio

5.2.2 Views

Once the perspective has been selected, the Design Studio interface appears as shown in Figure 5-3. The various panels are called *views*. A view is a visual component of the Design Studio that typically contains trees, access to editors, or a display of properties for the active editor. The most commonly used views in the BI perspective are the Data Project Explorer (upper left), Database Explorer (lower left), and Properties (lower right). The upper right view serves as an interactive canvas for creating objects such as *data flows* and *mining flows*, which are introduced later in this chapter.

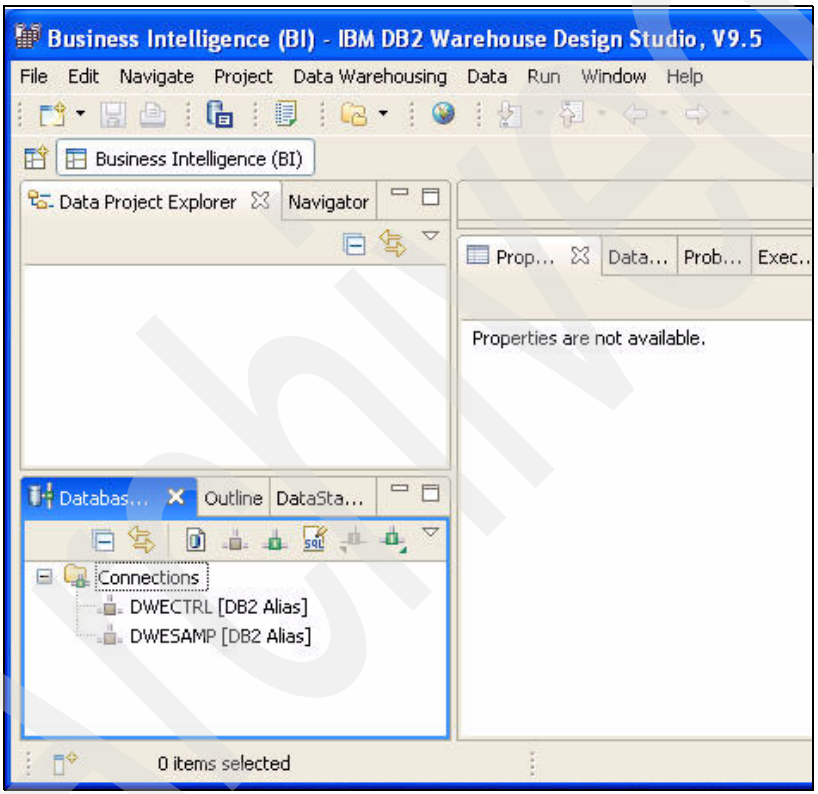


Figure 5-3 Design Studio: initial view with BI perspective

Data Project Explorer view

The Data Project Explorer view, located in the upper left of the BI perspective, is a view showing the files and metadata associated with data warehouse and data design *projects*. A project is an organizational tool in Design Studio, where different projects can be created for different purposes, such as multiple users or

teams. Projects can be exported from and imported into Design Studio within the same or a different workspace.

A project consists of a set of objects created as part of data transformation or warehouse building processes. Each project is represented by an icon or folder in the Data Project Explorer, where it can be expanded and its components accessed by editors. For data preparation and data mining, the *Data Warehouse Project* type is used. To create a new data warehouse project, the user selects **File** → **New** → **Project** from the menu bar. This brings up a New Project wizard, as shown in Figure 5-4, which enables the user to create a new project.

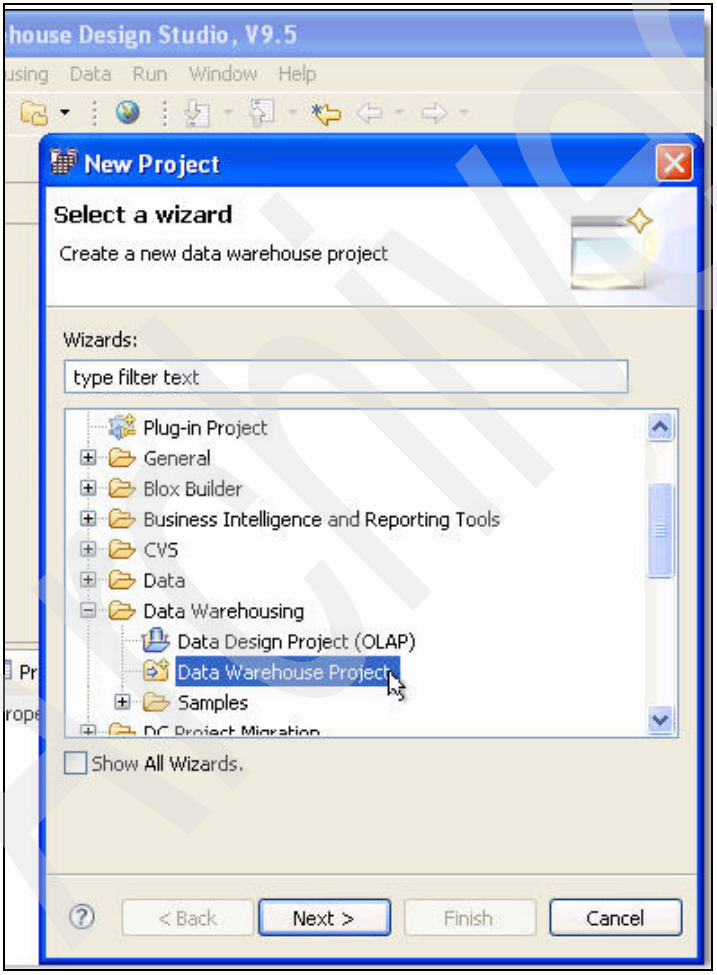


Figure 5-4 Design Studio: creating a new project

The new project is added to the Data Project Explorer, as shown in Figure 5-5 for a project called My Data Mining Project. The project contains objects such as Data Flows and Mining Flows that, in turn, contain objects used in performing data preparation and data mining processes.

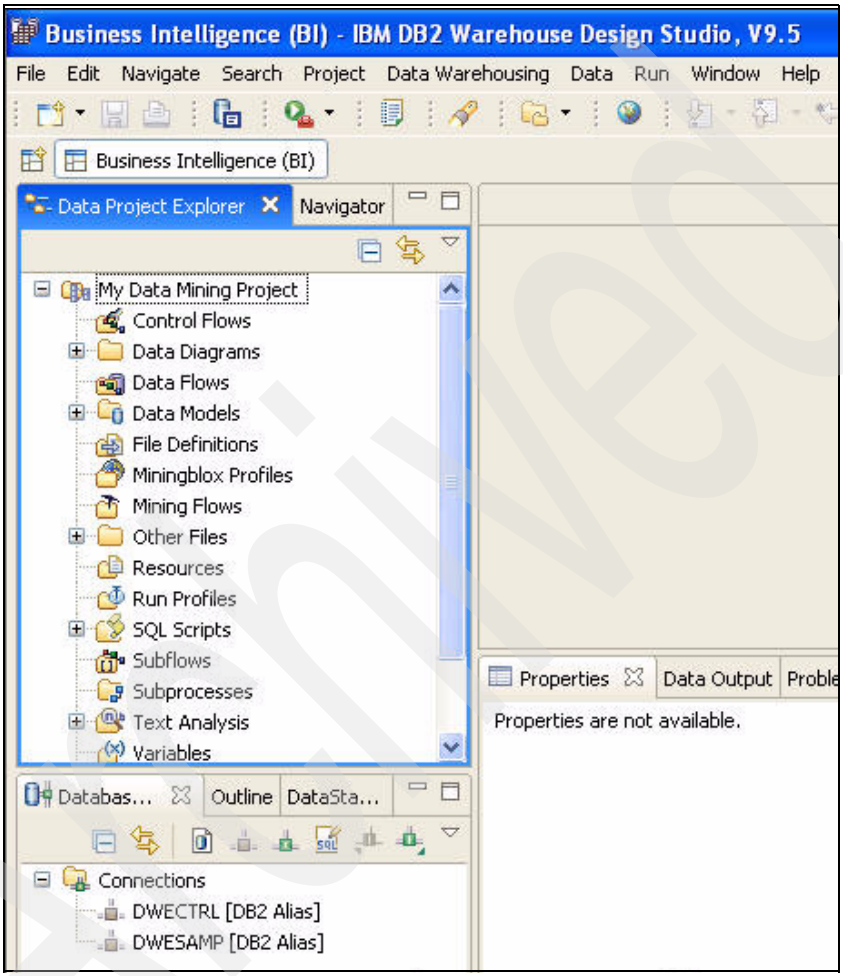


Figure 5-5 Design Studio: project view

Database Explorer view

The Database Explorer view is located in the lower left of the BI perspective. This view provides access to database connections and functions for data exploration, administration, and maintenance.

Database connections

Database connections are displayed in the Data Explorer view. Right-clicking a database or table object brings up a pop-up menu with functions relevant to that object, as illustrated in Figure 5-6 for a database.

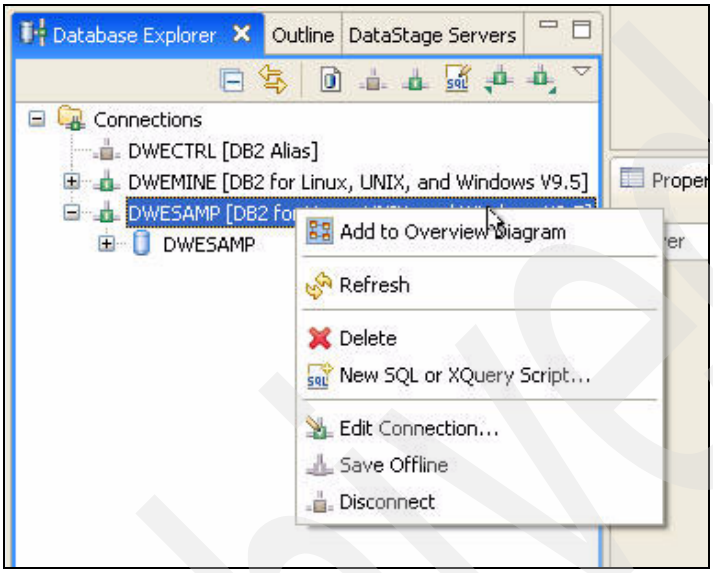


Figure 5-6 Database Explorer: database options

Functions are illustrated in Figure 5-7 for a table. A database connection can be refreshed by right-clicking the icon (Figure 5-6 on page 58) and refreshing the connection. Once a connection to a database is established, several database operations are available, such as altering or dropping a table, examining table contents, generating table DDL, and generating table statistics and distributions.

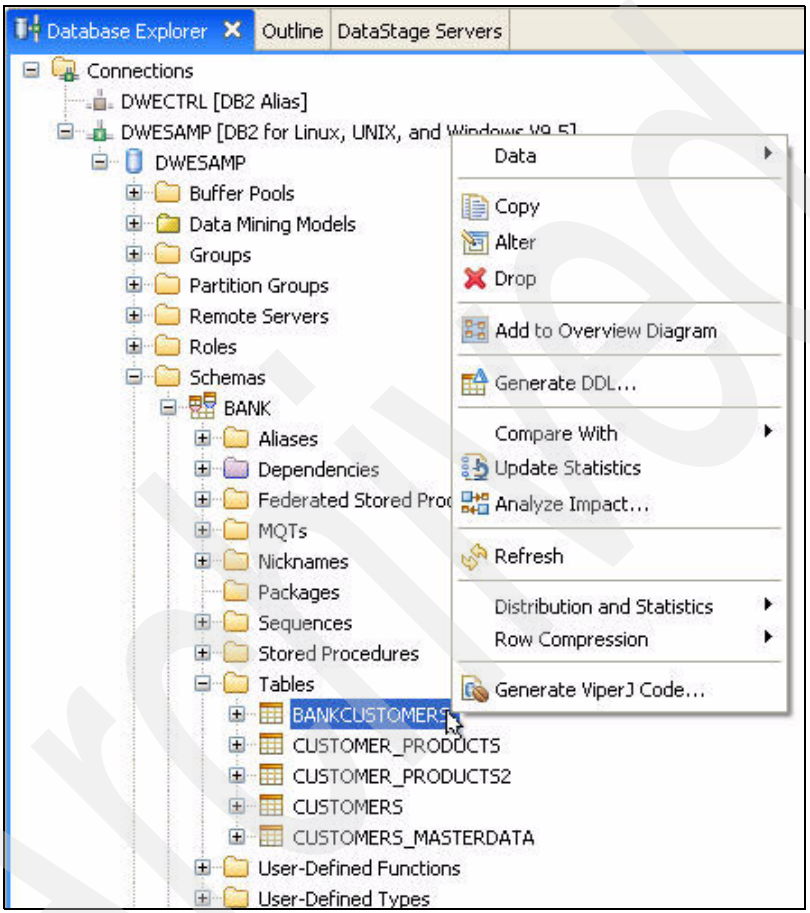


Figure 5-7 Database Explorer - table options

A new connection to an existing database can be created by right-clicking the **Connections** icon in the Database Explorer and selecting **New Connection** from the pop-up window, as shown in Figure 5-8.

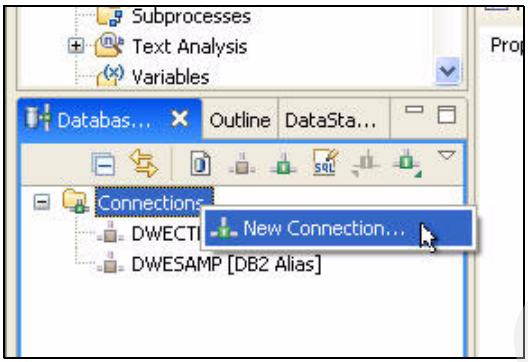


Figure 5-8 Design Studio: creating a new database connection

In the configuration wizard, as shown in Figure 5-9 on page 61, the analyst selects the appropriate database manager (DB2 for Linux®, UNIX®, or Windows, as in this case) and enters the database name, host name, port number, and authentication information. The new database connection then appears in the Connections list.

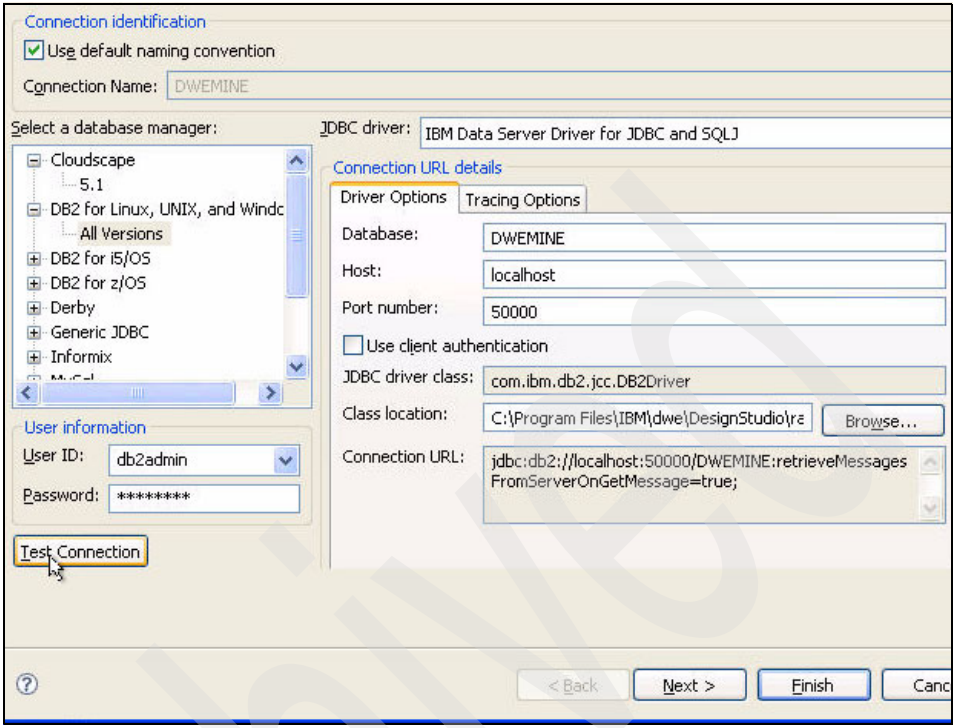


Figure 5-9 Database Explorer: configuring a new database connection

Database enablement

Each database used for data mining must be *enabled for data mining*, meaning that the appropriate DB2 extenders (kernel components for data mining) and other components are set up to enable data mining functionality. As illustrated in Figure 5-10 on page 62, a database is enabled for data mining by locating the database in the Connections folder, right-clicking the icon, and selecting the enablement option from the pop-up menu.

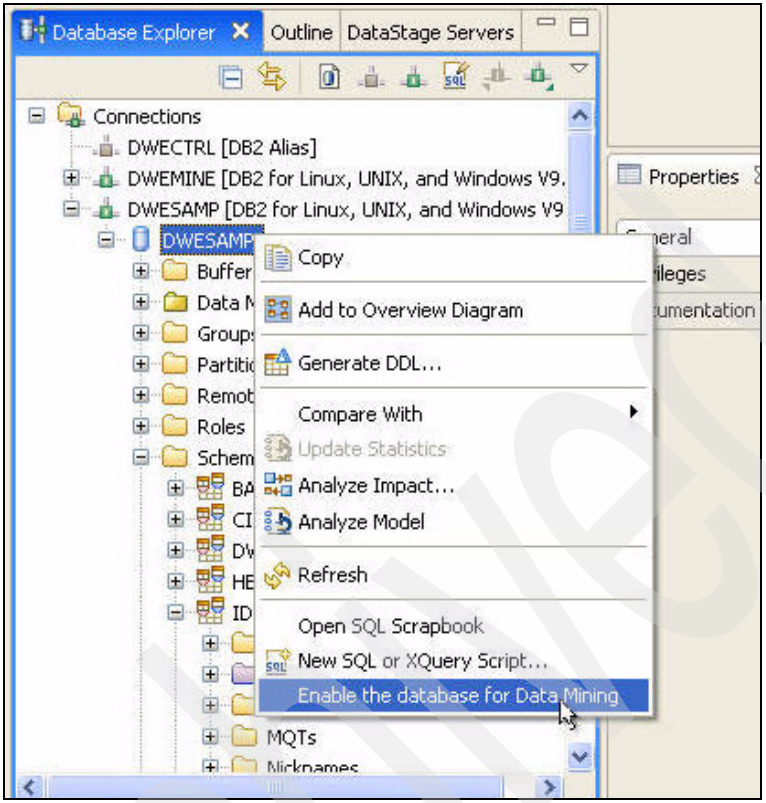


Figure 5-10 Database Explorer: enabling a database for data mining

When a database is enabled for data mining, the schema IDMMX and a set of tables used by the mining functions is automatically created. While most of these tables are repositories for settings and specifications created and used by the data mining procedures, four of them contain the data mining models stored in PMML (Predictive Modeling Markup Language) format:

- IDMMX.RULEMODELS: Associations and sequences models
- IDMMX.CLUSTERMODELS: Clustering models
- IDMMX.CLASSIFMODELS: Classification models
- IDMMX.REGRESSIONMODELS: Prediction models

Managing data mining models

Data mining models residing in the four mining models tables under the schema IDMMX can be opened, renamed, deleted, or exported in PMML format, as shown in Figure 5-11. A model can be exported and then imported into another workspace, or it can be exported to the local file system for exchange with another data mining tool.

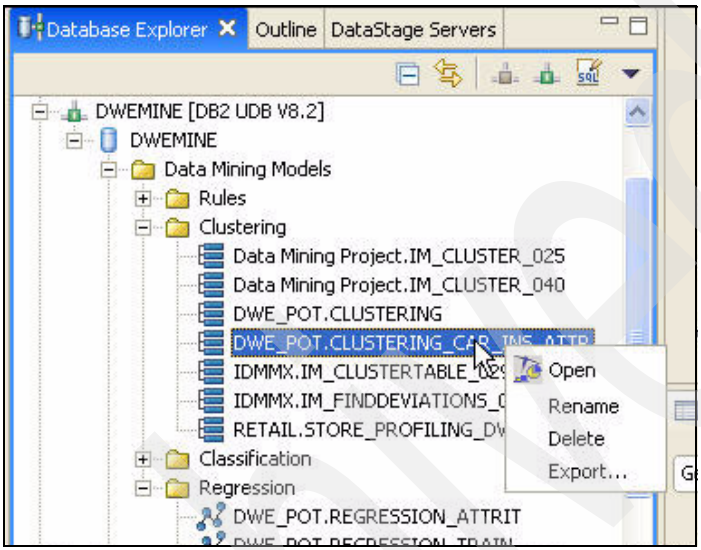


Figure 5-11 Database Explorer: managing data mining models

Data exploration

The Database Explorer in the DB2 Warehouse Design Studio offers several functions to examine the data in the database tables. Four functions allow the analyst to inspect data values and distributions quickly and easily through the Database Explorer. The following functions are fully described in Chapter 6, “Data preparation for data mining” on page 97:

- Table sampling
- Univariate distributions
- Bivariate distributions
- Multivariate distributions

Editor view

The Editor view, located in the upper right of the BI perspective, displays a canvas on which objects (*operators*) are placed, edited, and linked to construct data flows and mining flows. Double-clicking any operator on the canvas brings up a wizard for defining or modifying the properties of that operator. The Editor view includes a palette of operators available for use in flows. Design Studio provides a different editor for each of these object types:

- Physical data models
- Data flows
- Mining flows
- Control flows

Properties view

The Properties view, located in the lower right of the BI perspective, displays the properties of a selected object in the Editor view. The Properties view allows the analyst to view and modify the properties of the selected object.

5.3 Data flows

A data flow is a process containing a set of operations (such as transformations, derivations, calculations, aggregations, and joins) applied to source data (relational tables or flat files) to generate output data (one or more target tables). When creating a data flow, the developer imports the metadata (table names, schemas, column names) from the data sources to define components of the process. A data flow can have a live database connection during development of the flow.

5.3.1 Operators for data flows

Data flows are created in the Editor view of Design Studio by selecting operators from the palette and specifying properties for each operator. The operators represent the source, transform, and target steps in a data flow. The operators are graphical objects that are selected from the palette and dropped onto the canvas work area in the Editor view. The operators form the nodes in a data flow diagram. Each type of operator has a specific set of input and output data ports, one or more lists of data columns, and a number of properties that define how each operator moves or transforms data. The operators available on the palette for data flows are shown in Figure 5-12.

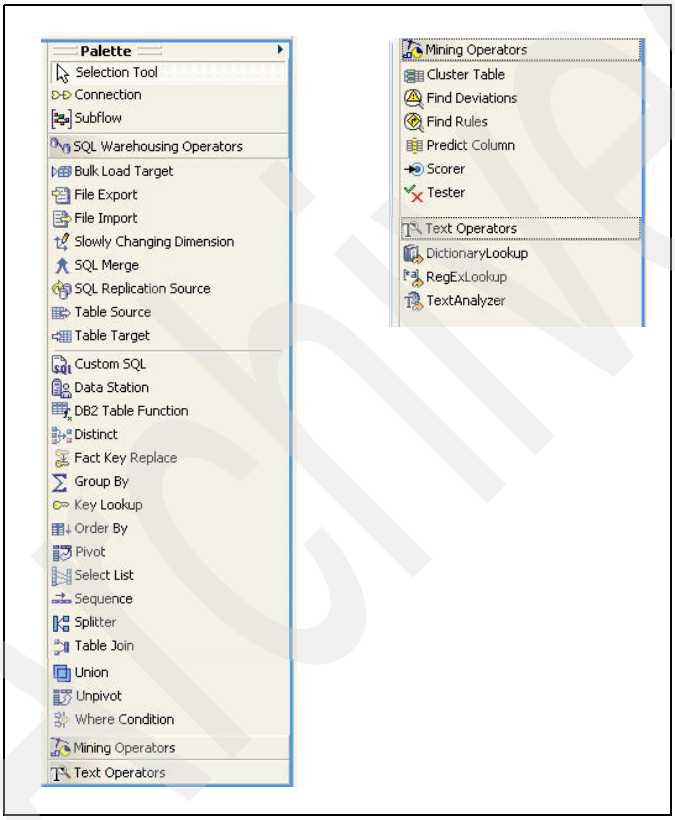


Figure 5-12 Editor view: operator palette for data flows

Operators are used in data flows as needed to build a desired process. All of the operators shown in Figure 5-12 on page 65 are discussed in the online documentation for DB2 Warehouse V9.5. The most common operators used in data flows are also available for mining flows. These operators are:

- Table source: Specifies an input table or view.
- Table target: Specifies an output table or view.
- Distinct: Select distinct values for a specified column.
- Group By: Performs an operation (such as sum and average) by values of one or more specified columns.
- Select List: Specifies a subset of columns.
- Table Join: Performs a join between two or more tables or views.
- Unpivot: Converts repeating values (such as months or years) in a single column, called the data group, as headings for a new set of columns.
- Where Condition: Filters records by specified data values.

Selected mining operators are available to data flows. These mining operators represent a subset of the mining functionality in mining flows and do not generate visualizations of mining results. They provide a means to generate certain mining results to feed into subsequent steps in the same data flow.

5.3.2 Data flow example

In the following example, we illustrate a simple data flow to join two data tables. More complex data flows that are used for data preparation for data mining are discussed in Chapter 6, “Data preparation for data mining” on page 97 and Chapter 9, “Solving a business problem with data mining” on page 237.

The data flow is shown in Figure 5-13. The operators are as follows:

- Operator 1: Specifies the first Table Source. This operator specifies a table of transactions for each of many stores.
- Operator 2: Specifies the second Table Source. This operator specifies a table of product information.
- Operator 3: Sets up the Table Join conditions to join the two tables sources.
- Operator 4: Sets up the Group By conditions for aggregating sales by store and department.
- Operator 5: Specifies the Target Table that receives the output records from the data flow operations.

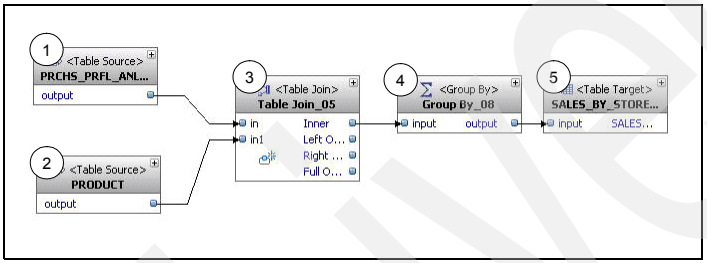


Figure 5-13 Data flow example

Details of the first Table Source operator (1) are shown in Figure 5-14. The analyst selects a Table Source operator from the palette and places it onto the canvas in the Editor view, then edits the properties of the operator to specify the desired table name, schema, and other properties of the source table. The second Table Source operator (2) is specified similarly.

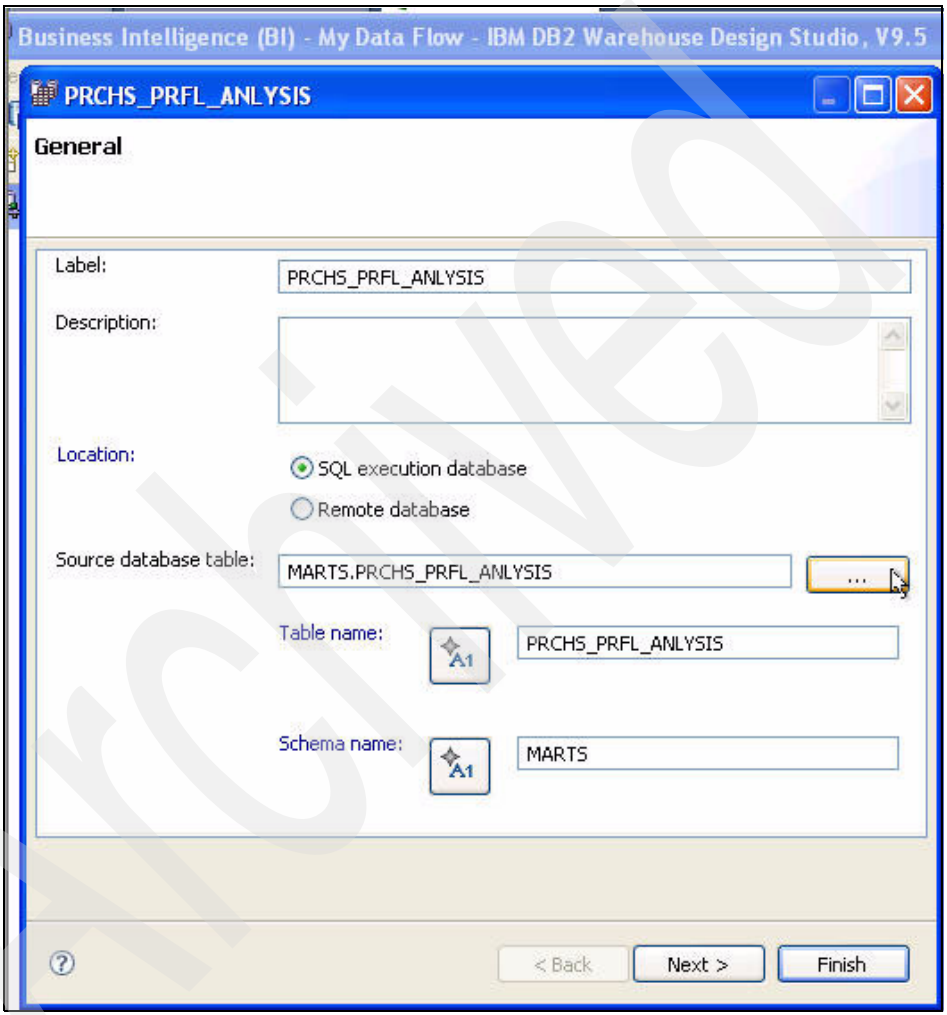


Figure 5-14 Data flow example: Table Source operator

The Table Join operator (3) in Figure 5-13 on page 67 is illustrated in Figure 5-15. Clicking the **...** button, as shown, accesses the expression builder to create the join condition.

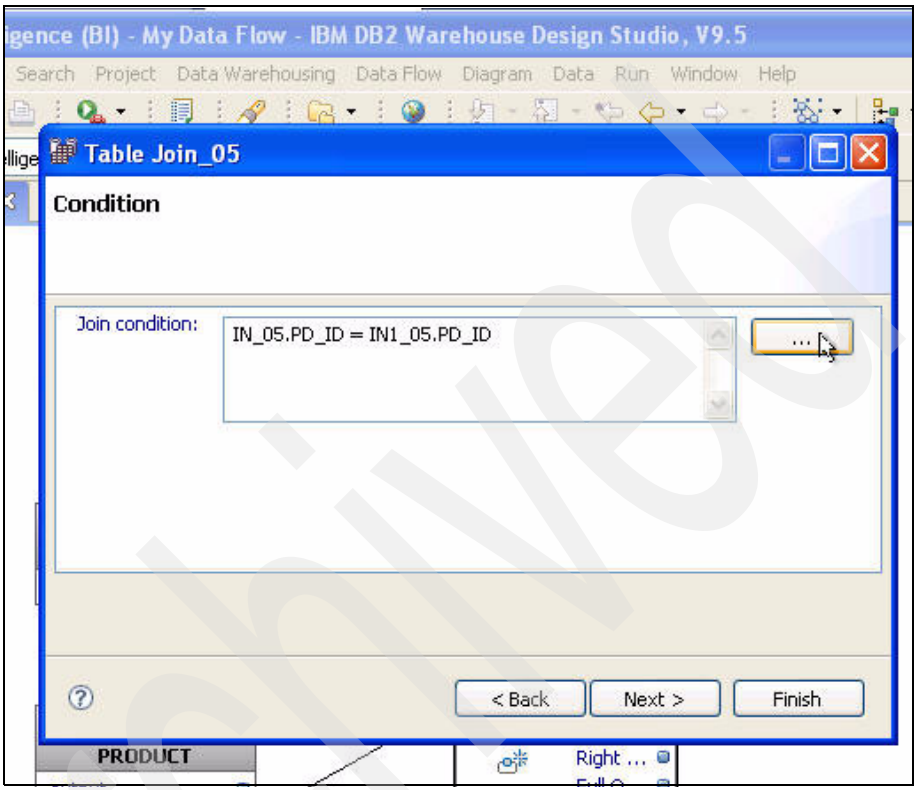


Figure 5-15 Data flow example: Table Join condition

The resulting columns from the table join are specified as shown in Figure 5-16. The analyst can type in customized column names if desired, as shown.

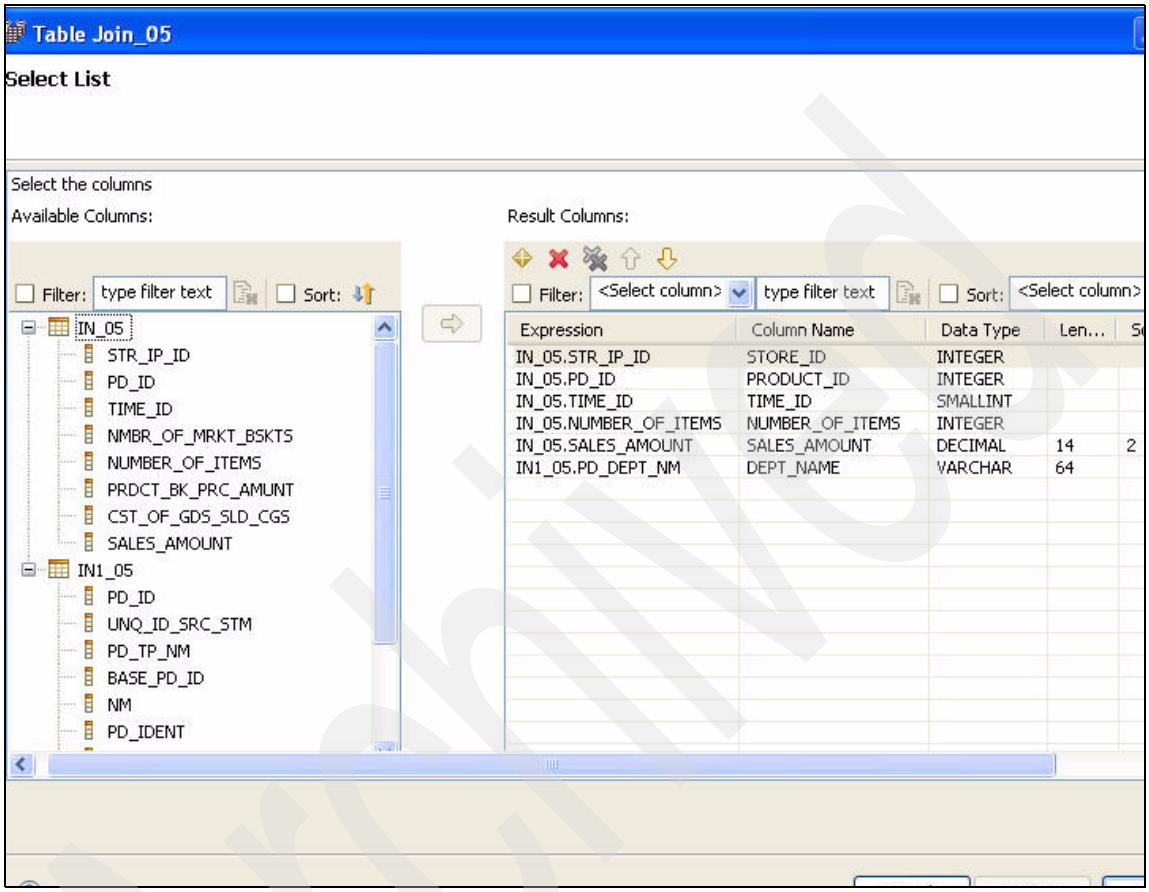


Figure 5-16 Data flow example: Table Join result columns

In the Group By operator (4) in Figure 5-13 on page 67, columns that are not needed can be highlighted and deleted, as illustrated in Figure 5-17.

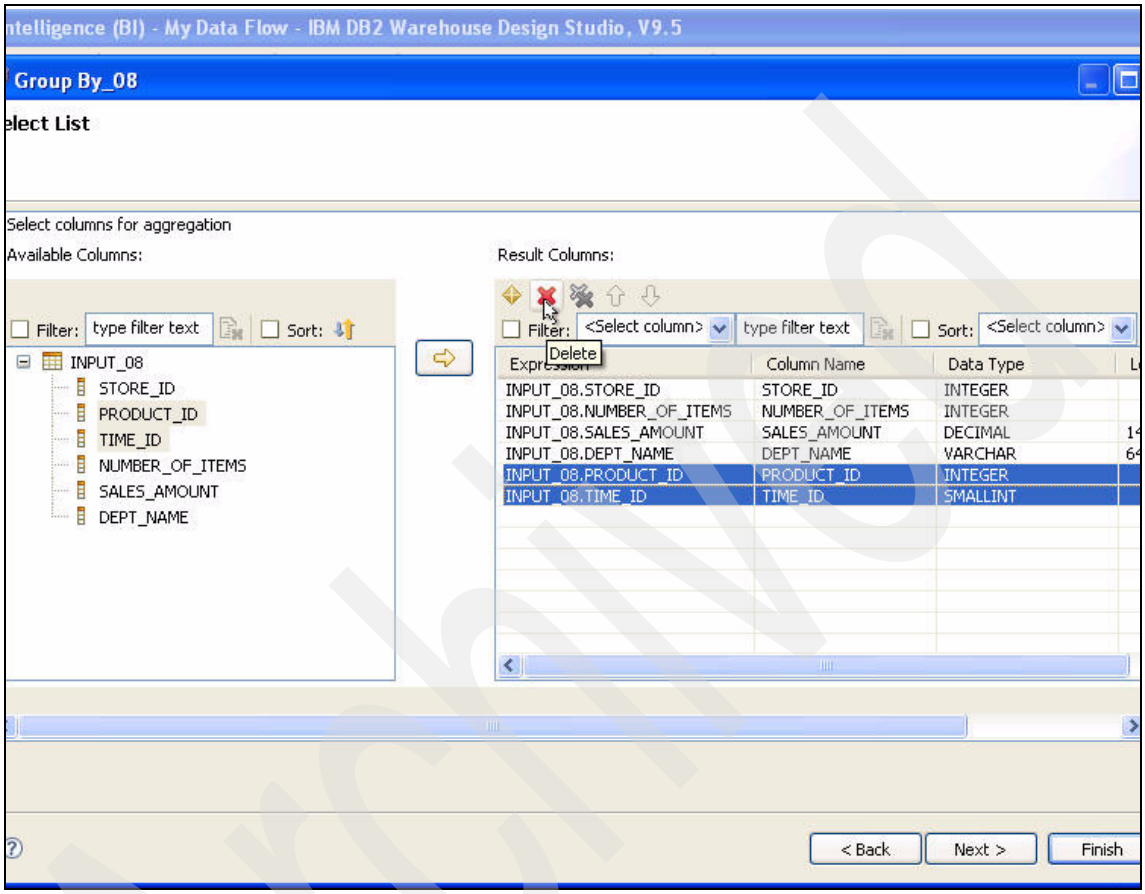


Figure 5-17 Data flow example: selecting Group By columns

In Figure 5-18, the resulting Group By columns result in aggregations of sales amount and number of items by store ID and department.

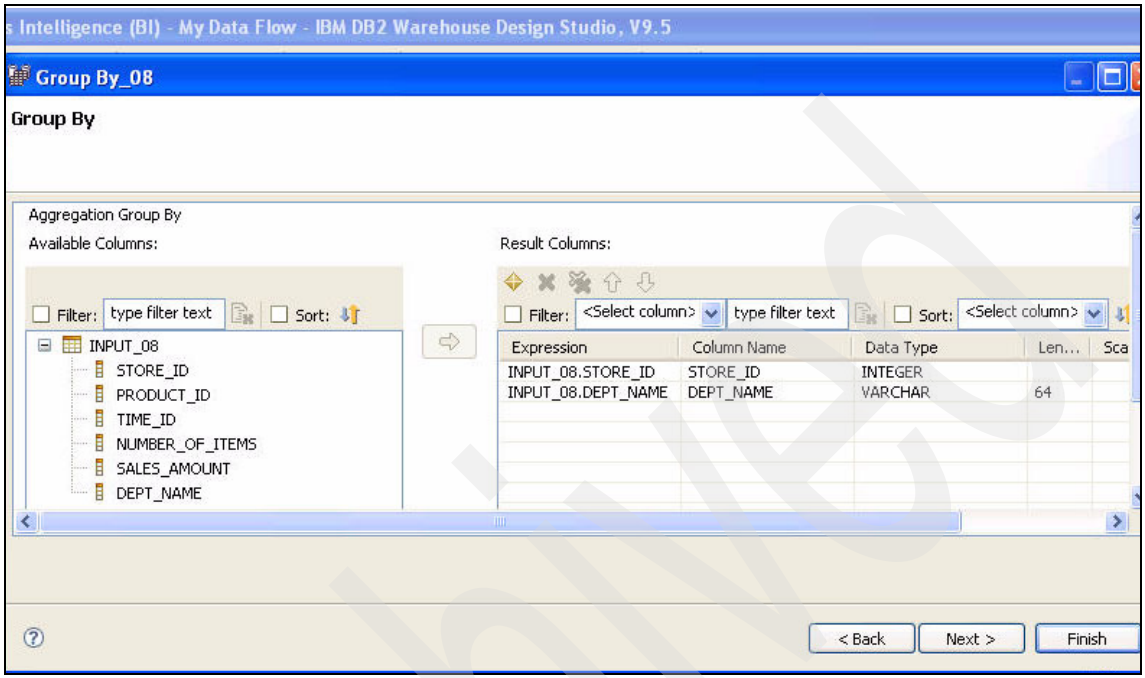


Figure 5-18 Data flow example: Group By columns selected

The Table Target operator (5) in Figure 5-13 on page 67 is created by right-clicking the output port of the Group By operator (4) and selecting **Create Suitable Table** from the pop-up menu, as shown in Figure 5-19. This function defines and creates a table to receive the result columns defined by the output operator (in this case, the Group By operator).

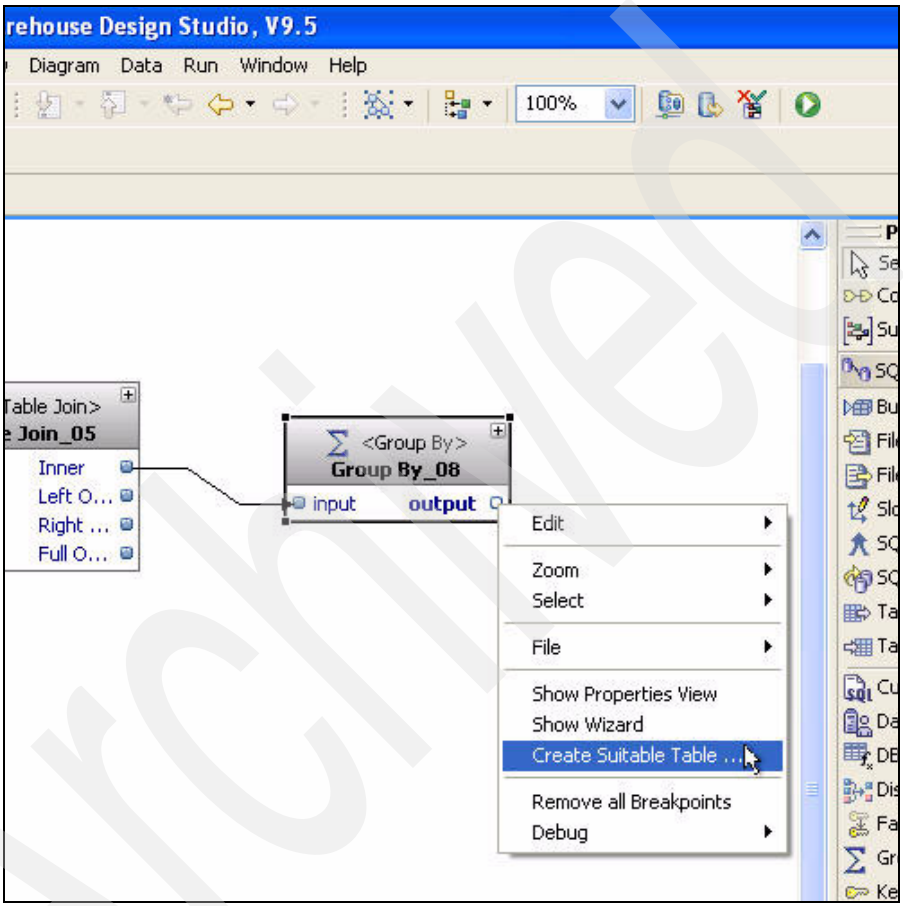


Figure 5-19 Data flow example - creating an output table

The analyst provides a name and other necessary information for the new table, as illustrated in Figure 5-20.

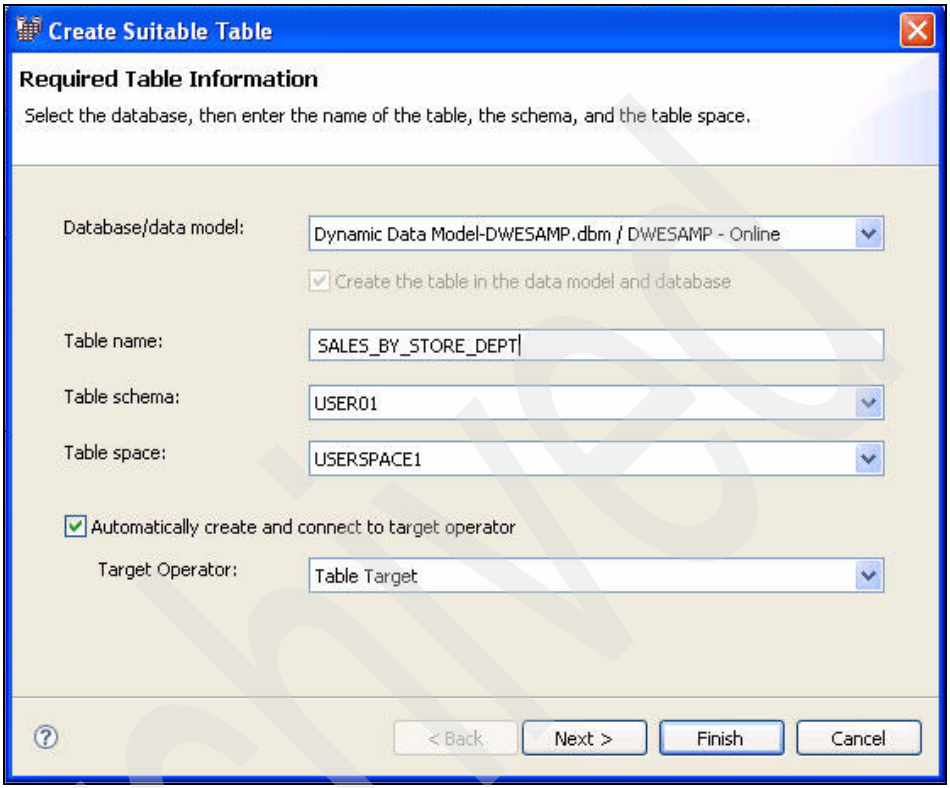


Figure 5-20 Data flow example: specifications for new output table

The completed data flow is executed either by selecting **Data Flow** → **Execute** from the menu bar or by clicking the **Execute** button, as shown in Figure 5-21. Executing the data flow causes all of the operations in the flow to be executed, resulting in the creation and population of the target table in this example.

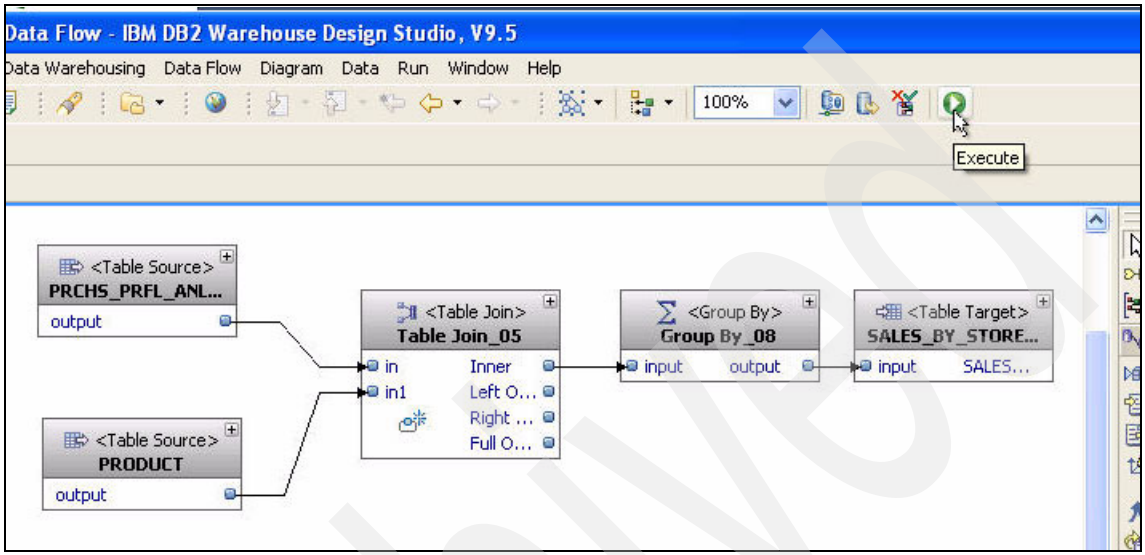


Figure 5-21 Data flow example: executing the data flow

The analyst can see the execution status and log, which is helpful in debugging if the execution fails, as illustrated in Figure 5-22.

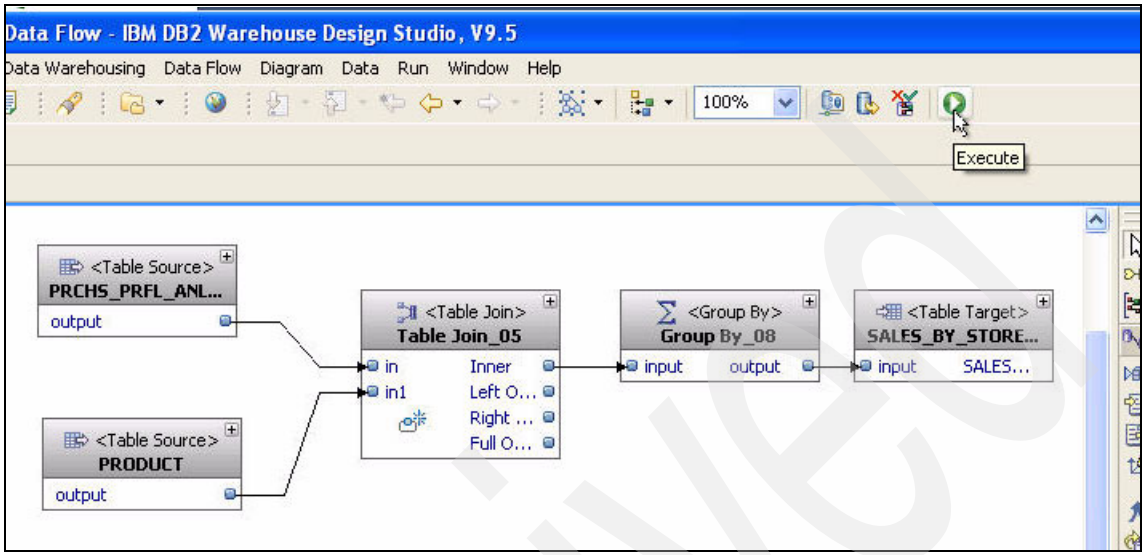


Figure 5-22 Data flow example: execution log and status information

5.4 Mining flows

A mining flow is a process containing a set of data mining operations applied to source data to build and test data mining model(s), generate mining results, and output one or more target tables containing mining results and selected source data. The operations may include data preparation as well as data mining functions. A mining flow has a live database connection during development of the flow.

5.4.1 Operators for mining flows

Like data flows, mining flows are created in the Editor view of Design Studio by selecting operators from the palette and specifying properties for each operator. The operators for mining flows include those for data flows plus additional operators specific to mining. These operators fall into the categories of sources and targets, preprocessing, and mining. The operators available on the palette for mining flows are shown in Figure 5-23.

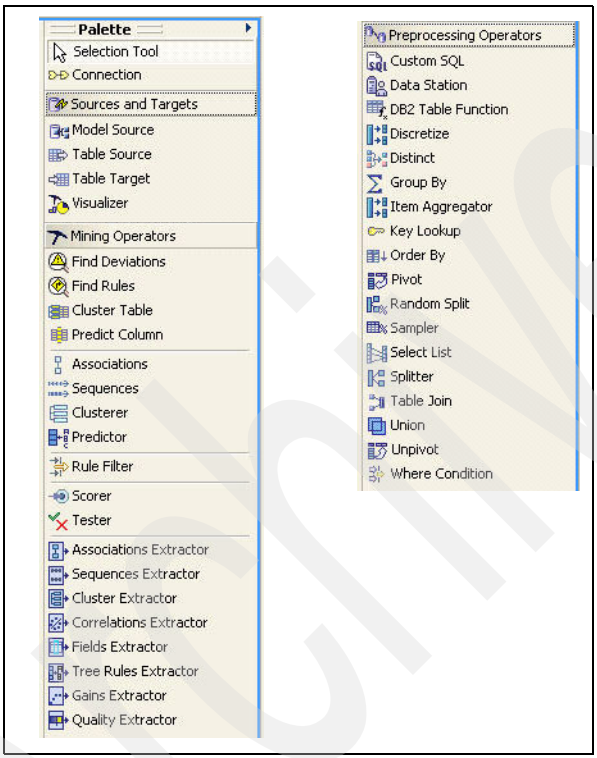


Figure 5-23 Editor view: operator palette for mining flows

Preprocessing operators

Most of the preprocessing operators in Figure 5-23 are also included in the data flow operator palette in Figure 5-12 on page 65. For mining flows, there are also some mining-specific preprocessing operators:

- Discretize: Categorizes values of a numeric variable into bins.
- Random Split: Randomly divides source data into two mutually exclusive sets for training and testing mining models.
- Sampler: Extracts a random sample from source data.

Sources and targets operators

Source and targets operators define the inputs and outputs in mining flows. They are:

- Visualizer: Displays graphical and tabular information about the results of a data mining model. The visualizer presents results different content in different formats, depending on the type of mining model.
- Model Source: Specifies a data mining model (previously created and stored in the appropriate table) as a source for scoring and extractor operators.

Mining operators

The mining operators perform mining tasks such as executing mining algorithms (clustering or associations), extracting mining results, such as associations rules or cluster attributes, visualizing mining models, and applying mining models to score new records. The mining operators are:

- Find Deviations: Finds records in a source table or view that deviate from the majority of records, based on clustering, such as potentially fraudulent behavior by customers who consistently have high numbers of accident claims. This is a high-level operator that needs only minimal parameter specifications and does not use a Visualizer operator. It creates a clustering model that can be viewed by opening it in the Database Explorer view.
- Find Rules: Finds affinity relationships in a source table or view and creates a set of associations rules describing those relationships. This operator accepts data in either transactional or behavioral-demographic format. This is a high-level operator that uses minimal parameter specifications and does not use a Visualizer operator. It creates a rule model that can be viewed by opening it in the Database Explorer.
- Cluster Table: Finds segments with similar characteristics in a source table or view. This is a high-level operator that uses minimal parameter specifications and does not use a Visualizer operator. It creates a clustering model that can be viewed by opening it in the Database Explorer.
- Predict Column: Predicts values of a target field by building a classification or regression model from the columns in a source table or view. This is a high-level operator that uses minimal parameter specifications and does not use a Visualizer operator. The classification or regression model can be viewed by opening it in the Database Explorer.
- Associations: Builds an associations rule model containing affinity rules among items in a transactional source table or view. The user has extensive control over parameters. The operator sends model results to a Visualizer operator, an Associations Extractor operator, or a Table Target operator.

- Sequences: Builds a sequences rule model containing sequences of item affinities across transactions in a transactional source table or view. The user has extensive control over parameters. The operator sends model results to a Visualizer operator, a Sequences Extractor operator, or a Table Target operator.
- Clusterer: Builds a clustering model that contains segments of similar records from a source table or view. The user has extensive control over parameters. The operator sends model results to a Visualizer operator, a Cluster Extractor operator, or a Table Target operator.
- Predictor: Creates a mining model that predicts the value of a target field in a training data set generated by a Random Split operator. Depending on whether the target field is categorical or numeric, the resulting model type is either classification or regression, respectively. The user has extensive control over parameters. The operator sends model training results to a Visualizer operator, an extractor operator, or a Table Target operator.
- Tester: Tests a classification or regression model by applying the model to a testing data set generated by a Random Split operator. The operator computes test results containing information about model quality and performance. The operator sends model testing results to a Visualizer operator or a Table Target operator.
- Scorer: Applies a data mining model to a source table to compute scores for new records using a model generated by an Associations, Sequences, Clusterer, or Predictor operator or a model created by a third-party tool and imported into DB2 Warehouse through the Database Explorer view. The operator sends scoring results and, depending on user-specified settings, some or all of the source data to a Table Target operator.
- Associations Extractor: Extracts information from an associations rules model and sends the extracted information to a Table Target operator.
- Sequences Extractor: Extracts information from a sequences rules model and sends the extracted information to a Table Target operator.
- Cluster Extractor: Extracts information from a clustering model and sends the extracted information to a Table Target operator.
- Correlations Extractor: Extracts pairwise correlations from a clustering, classification, or regression model and sends the extracted information to a Table Target operator.
- Fields Extractor: Extracts field information (such as the input fields used in the model and the importance of each field in the model) from a mining model and sends the extracted information to a Table Target operator.
- Tree Rules Extractor: Extracts information (including decision paths, node ID, and predicted values, and confidence) from a decision tree classification model and sends the extracted information to a Table Target operator.

- Gains Extractor: Extracts a gains chart information from a classification or regression model and sends the extracted information to a Table Target operator.
- Quality Extractor: Extracts model quality information (specific to the model type) from a data mining model or a test result and sends the extracted information to a Table Target operator.

5.4.2 Mining flow example

In the following example, we illustrate a simple mining flow to create a clustering model and output the records and their respective best-fit cluster information to a table. The mining flow is shown in Figure 5-24. The operators are as follows:

- Operator 1: Specifies the Table Source. This operator specifies a table of heart patient data with one record per patient.
- Operator 2: Sets up the clustering model and specifies parameters and setting to create the model.
- Operator 3: Pulls in the mining model results and displays them in a set of figures and tables.
- Operator 4: Applies the model from (3) to the source records in (1), scores the patients to assign them to their respective best-fit clusters, and writes out the scoring results and user-selected input data to a Target Table.
- Operator 5: Specifies the Target Table that receives the output records from the Scorer (4).

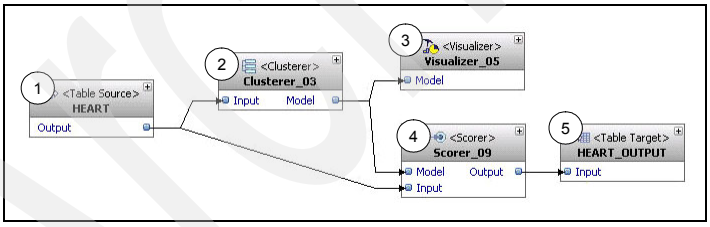


Figure 5-24 Mining flow example

Properties of the Clusterer operator are shown in Figure 5-25. For the clustering method, the analyst can choose from two algorithms in the **Algorithm** drop-down menu.

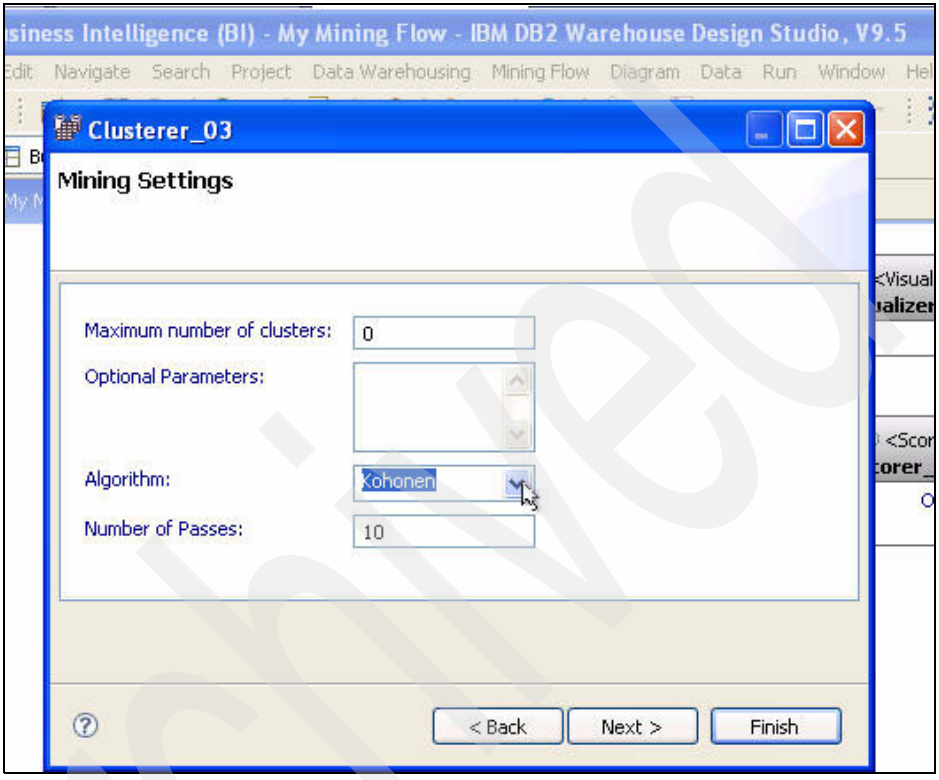


Figure 5-25 Mining flow example: Clusterer operator

Properties of the Scorer operator are shown in Figure 5-26. Scoring results (result columns) to be written to the output table are specified using the **Add**, **Edit**, and **Delete** buttons.

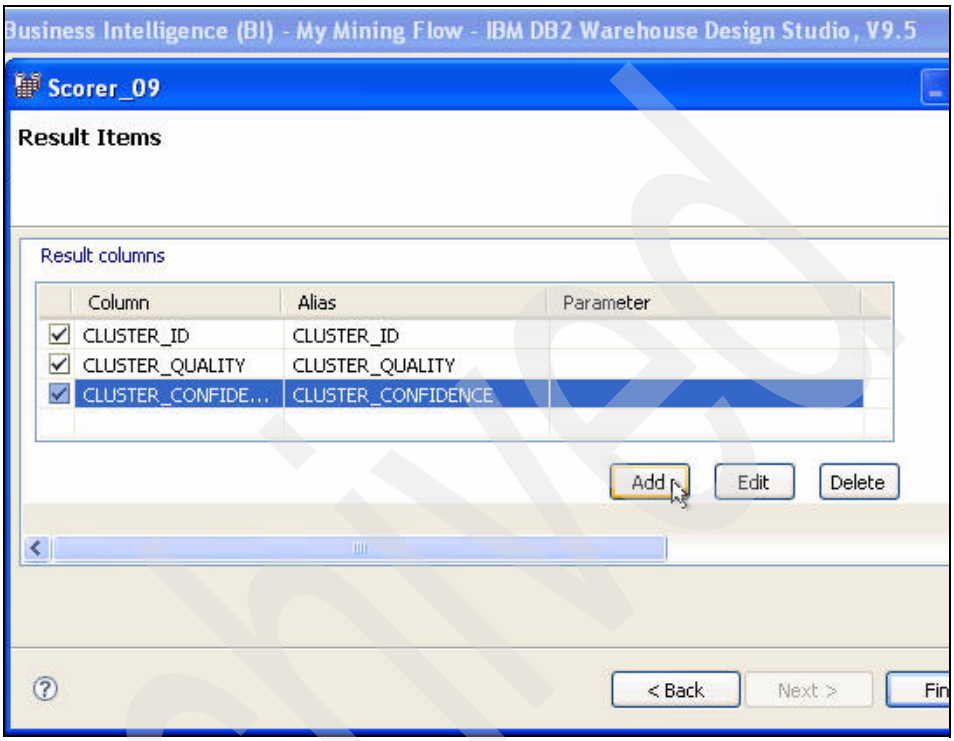


Figure 5-26 Mining flow example: Scorer operator

5.4.3 Scoring using third-party PMML models

Many data mining models created by and exported from third-party tools in PMML format can be imported into DB2 Warehouse and used for scoring source tables. This capability enables high-volume, high-speed, parallelized scoring of data in DB2 using third-party models.

A PMML model is imported into DB2 Warehouse through the Database Explorer view, as shown in Figure 5-27.

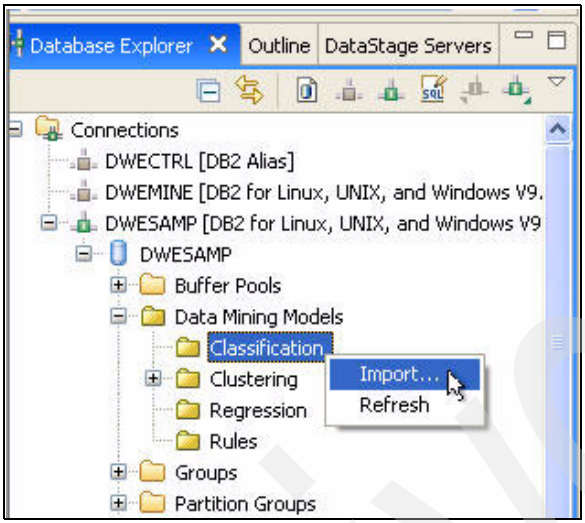


Figure 5-27 Importing a PMML model through the Database Explorer

Right-clicking the appropriate **Data Mining Models** folder and selecting **Import** from the pop-up window brings up a wizard for selecting the model to be imported, as shown in Figure 5-28.

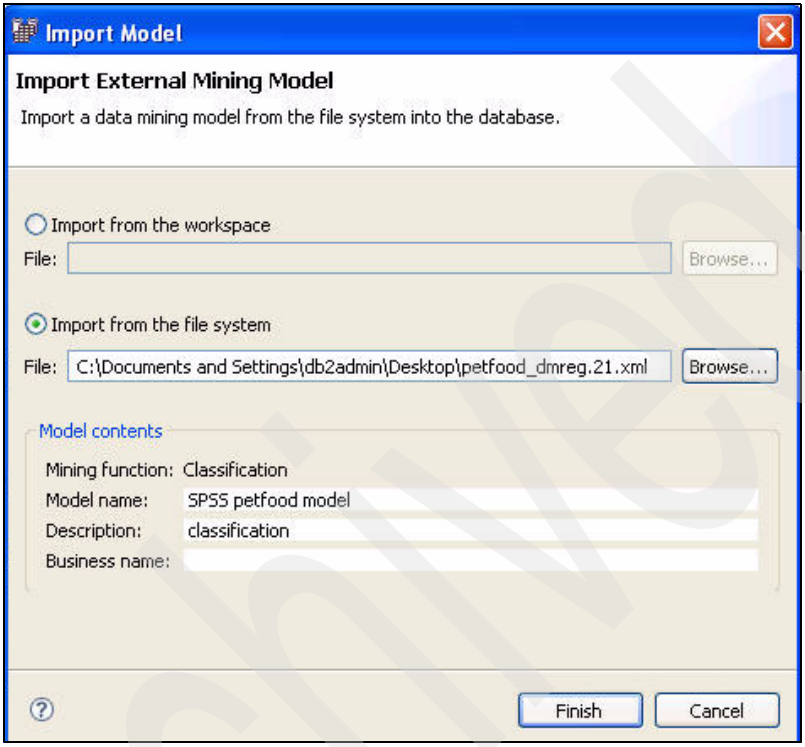


Figure 5-28 Importing a PMML model

The mining flow to use the imported model for scoring a source table is shown in Figure 5-29. A Model Source operator specifies the model, and a Table Source operator specifies the table to be scored. The Scorer operator applies the model to the data and sends the scoring results and selected input data to a Table Target operator as shown.

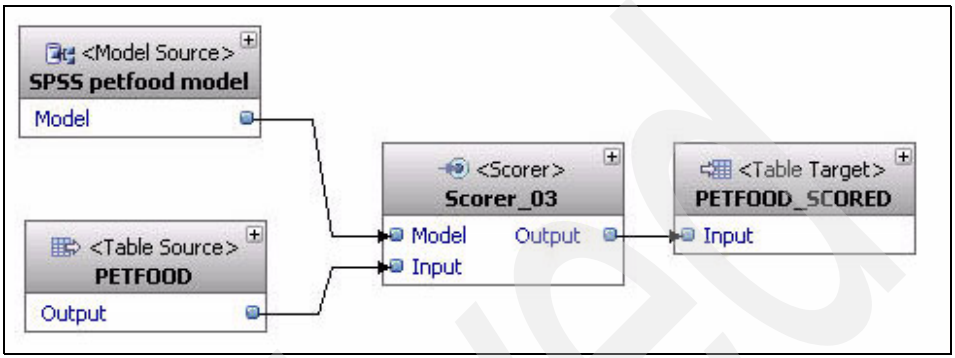


Figure 5-29 Mining flow for scoring using a PMML model source

5.5 Tooling for building data mining solutions

Two categories of tooling can be used to build a data mining solution in a DB2 data mining environment. These include tooling for data preparation and data mining, and tooling for data mining alone.

5.5.1 Tooling for data preparation and data mining

Tooling for data preparation and data mining is driven primarily by the needs of the data mining expert. These expert users may require complex data preparation and data mining techniques to solve complex business problems. Supporting processes may include automated data cleansing and preparation procedures embodied in the database (such as project- or domain-specific data marts), real-time data streaming, and statistical analysis. Integral activities may include data exploration, data refinement, model validation, and model tuning.

DB2 Warehouse Design Studio

DB2 Warehouse Design Studio provides capabilities and functionality for data preparation, data exploration, data mining model development, and in-database scoring using DB2-generated mining models or PMML models from other tools. Design Studio is an integrated framework for all of these capabilities.

Data mining workbenches

Data mining workbenchs generally provide functionality for data preparation, data exploration, statistical analysis, data mining model development, and scoring. They may work directly against a DB2 database, or they may extract data from DB2 or other sources into their own repositories. They may be able to export data mining models in PMML format, which can be imported into DB2 for in-database scoring.

SQL scripts

DB2 SQL scripts can be used in batch processes for data preparation or to invoke DB2 mining procedures for data mining model development and in-database scoring. These scripts can be integrated with existing or new processes used to build and maintain the information warehouse.

5.5.2 Tooling for on demand data mining

Tooling for on demand data mining is driven primarily by the needs of the business user. On demand data mining means that business users can build, refresh, visualize, and apply data mining models in their daily activities. These users perform high-value data mining that usually can be accomplished using less complex data mining techniques. For example, category managers in a retail corporation may use an application to regularly perform store segmentation at the category level to help them optimize product allocation and assortments among stores.

BI analytical tools

DB2 data mining procedures can be integrated into BI analytical tools to enable business users to build, refresh, visualize, and apply data mining models. For example, many IBM partners such as Business Objects, Cognos, and Microstrategy have integrated DB2 data mining procedures into their widely-used analytical and reporting tools. This integration extends the capabilities of those tools to include data mining for business users already familiar with them.

Web-based analytical tools

DB2 data mining procedures can also be integrated into Web-based analytical applications. A Web application can be designed as an interface for a business analyst to access data mining functions for creating, refreshing, or applying mining models.

DB2 Alphablox

DB2 Alphablox is a component of DB2 Warehouse for creating Web-based analytical applications or embedding data mining functionality in an existing Web application. It provides a robust set of functions to facilitate development of analytical applications. These applications can be highly customized and tailored to specific users' needs. For example, different users of one application may have access to specific content and functionality related to their respective roles. An example of a DB2 Alphablox Web-based data mining application is shown in Figure 5-30, which executes associations analysis to find item affinities for a market basket analysis. The business user can specify data selection criteria and data mining parameters used to build the data mining model.

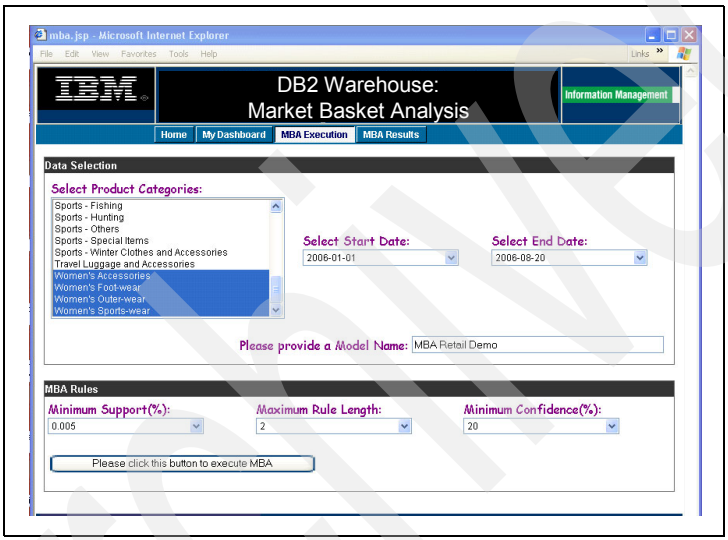


Figure 5-30 DB2 Alphablox Web-based data mining application: model building

5.6 Tooling for deploying data mining solutions

Two categories of tooling can be used to deploy a data mining solution into a business process in a DB2 environment. These include process-driven tooling and business user-driven tooling.

5.6.1 Tooling for process-driven deployment

Process-driven tooling includes DB2 Warehouse Design Studio and DB2 SQL scripts. This tooling enables refreshing of data mining models and extraction of data mining results for further utilization in business processes.

DB2 Warehouse Design Studio

DB2 Warehouse Design Studio allows the creation of control flows to represent a sequence of SQL steps. These steps may include data flow(s) for data preparation and mining flow(s) for data mining processing (model refreshing, scoring, and extraction of results). The control flow process can be automated in the database to ensure that the most recent information is available for on demand or real-time analytical needs, such as automated updating of customer information.

The output of a control flow is one or more tables or files that are used as input to another process or application. For example, a control flow may be executed to score a set of prospects by their propensity to respond to a promotion, in support of a marketing campaign. As another example, a control flow may be executed periodically to score a set of insurance customers by their expected loss ratio (claims paid out relative to premiums collected) for use in actuarial analysis.

SQL scripts

DB2 SQL scripts can be used in batch processes to deploy data mining processing for model refreshing, scoring, and extraction of results. These scripts can be automated in the database to ensure that the most recent information is available for on demand or real-time analytical needs. Similar to a control flow, the output of SQL scripts is one or more tables or files that are used as input to another process or application.

5.6.2 Tooling for business user-driven deployment

Business user-driven tooling includes BI analytical tools and Web-based analytical or transactional applications to deploy data mining to large numbers of business users and to deliver the right information to those who need it at the time they need it to perform their business activities. This tooling enables refreshing of data mining models, integration of data mining results with reporting and online analytical processing (OLAP), and real-time scoring.

BI analytical tools

BI analytical tools can execute data mining procedures to enable business users to refresh, visualize, and apply data mining models and integrate data mining results into operational and analytical applications such as *ad hoc* reporting or dashboards. The same IBM Business Partner tools used for building mining solutions can also be used to deploy mining solutions.

Web-based analytical tools

Web applications can be tailored to allow different levels of access or content to different users, depending on their job functions. For example, some business analysts may be able to refresh data mining models, while others may be able to apply those models for scoring. Another type of user may view mining results as part of a portal-based dashboard.

DB2 Alphablox

Highly customized DB2 Alphablox Web applications can be used to deploy data mining results as well as create data mining models. These applications can be tailored to specific users' needs, delivering the information that they need in their daily activities. Execution and visualization of data mining results from the DB2 Alphablox Web-based data mining application (Figure 5-30 on page 87) are shown in Figure 5-31.

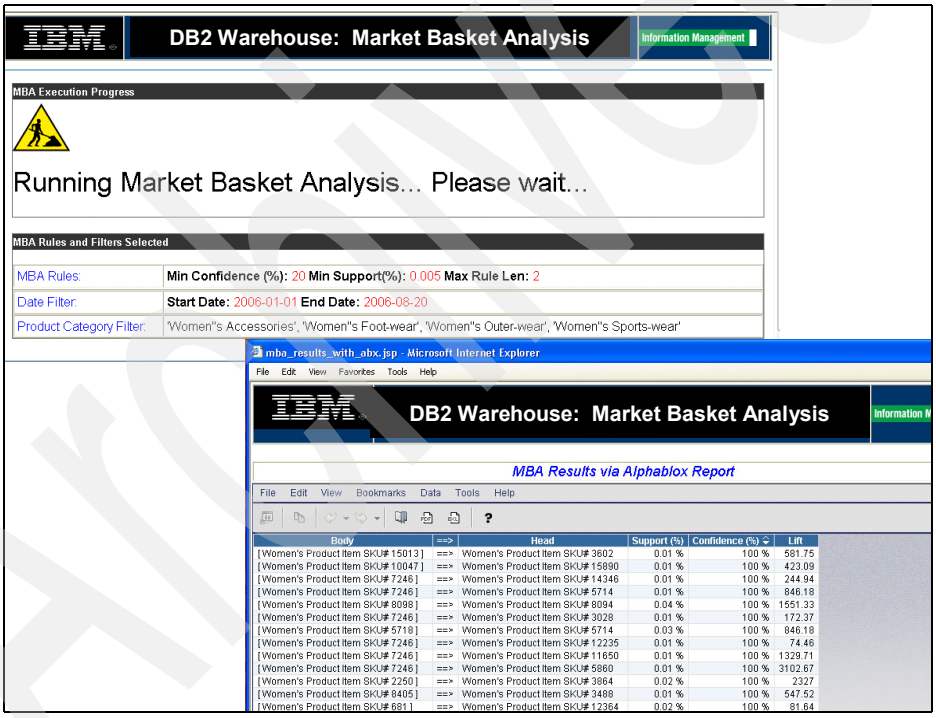


Figure 5-31 DB2 Alphablox Web-based data mining application: execution and results

DB2 Miningblox

DB2 Miningblox is an extension to DB2 Alphablox to provide data mining-specific functionality. This extension facilitates the creation of Web-based analytical applications to deliver data mining solutions. It provides the following functionality:

- Specify data input criteria for the data mining process.
- Execute the data mining process to refresh mining models or score records.
- Display mining results graphically in a Web browser.

Any DB2 Alphablox application using DB2 Miningblox for mining functions requires the following:

- A data warehouse application (deployment package for SQL scripts such as a control flow) containing prebuilt mining flows, previously deployed on the DB2 Warehouse Administration Console
- Miningblox tags to execute the data warehouse application and display the mining results

Unlike a purely DB2 Alphablox application, which can issue calls to DB2 mining procedures for model building, a DB2 Alphablox application using DB2 Miningblox for its mining functionality has no means to build a data mining model. For the latter, a data mining model has to pre-exist in a Design Studio mining flow and be deployed in a DB2 Warehouse data warehouse application, although the model can be refreshed through the DB2 Miningblox-based application.

A DB2 Miningblox application is built in the BI perspective of the Design Studio. This application requires that a control flow containing an appropriate data flow or mining flow already exists. Similar to a data flow or mining flow, a wizard guides the building of a DB2 Miningblox application. This wizard generates templates that the user can customize. An example of an application template is shown in Figure 5-32.

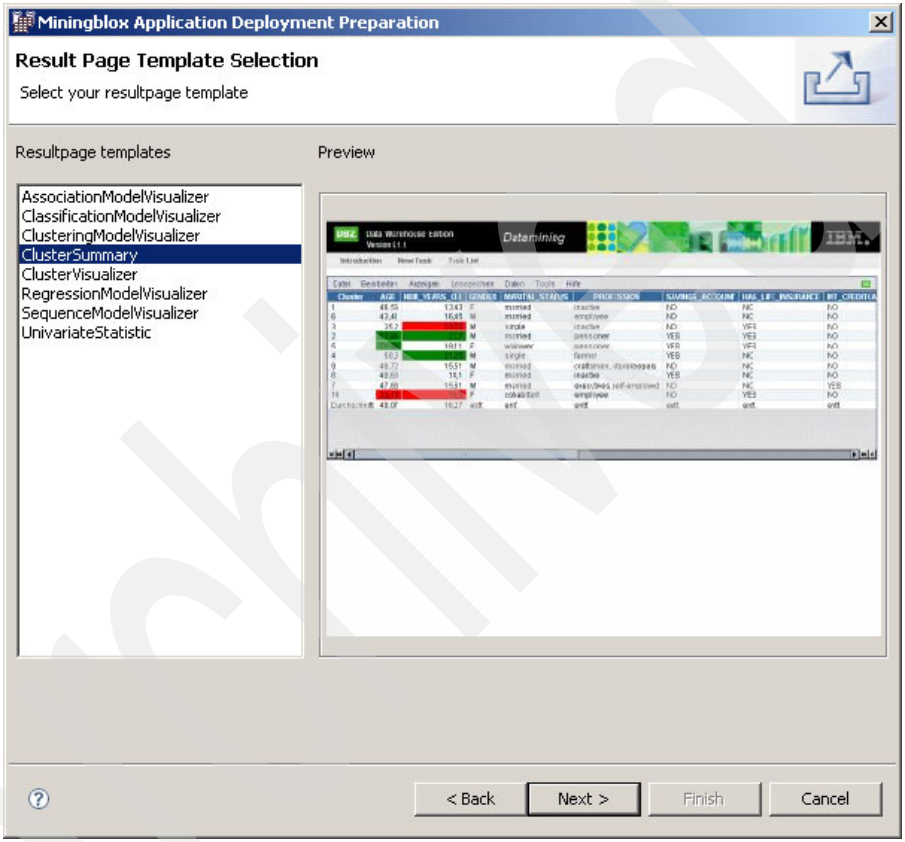
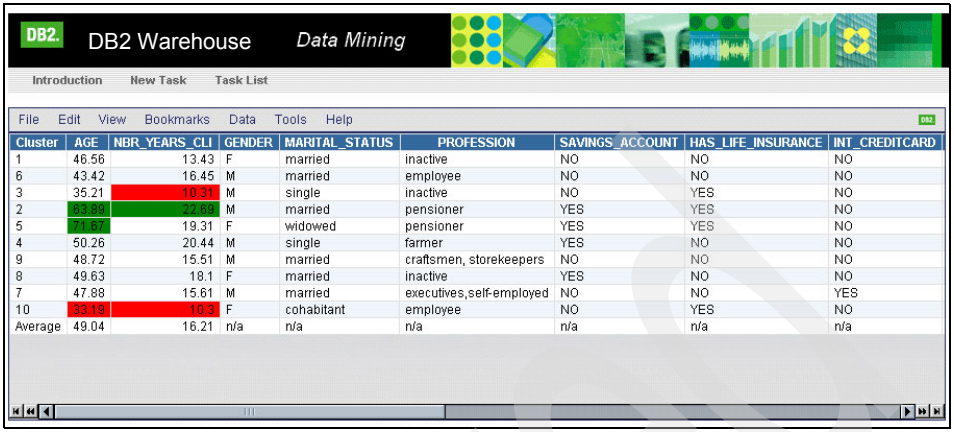


Figure 5-32 DB2 Miningblox application template

Mining results corresponding to this template are shown in Figure 5-33.



Cluster	AGE	NBR_YEARS_CLI	GENDER	MARITAL_STATUS	PROFESSION	SAVINGS_ACCOUNT	HAS_LIFE_INSURANCE	INT_CREDITCARD
1	46.56	13.43	F	married	inactive	NO	NO	NO
6	43.42	16.45	M	married	employee	NO	NO	NO
3	35.21	10.31	M	single	inactive	NO	YES	NO
2	63.89	22.69	M	married	pensioner	YES	YES	NO
5	71.67	19.31	F	widowed	pensioner	YES	YES	NO
4	50.26	20.44	M	single	farmer	YES	NO	NO
9	48.72	15.51	M	married	craftsmen, storekeepers	NO	NO	NO
8	49.63	18.1	F	married	inactive	YES	NO	NO
7	47.88	15.61	M	married	executives,self-employed	NO	NO	YES
10	33.19	10.3	F	cohabitant	employee	NO	YES	NO
Average	49.04	16.21	n/a	n/a	n/a	n/a	n/a	n/a

Figure 5-33 DB2 Miningblox application: visualization of mining results

5.7 Mining model management using Admin Console

The DB2 Warehouse Administration Console (Admin Console) is a Web-based application for managing and monitoring BI applications in the DB2 data warehouse environment. For data mining, the most important functions of the Admin Console are for model management and caching. The main view of the Admin Console is shown in Figure 5-34 on page 93.

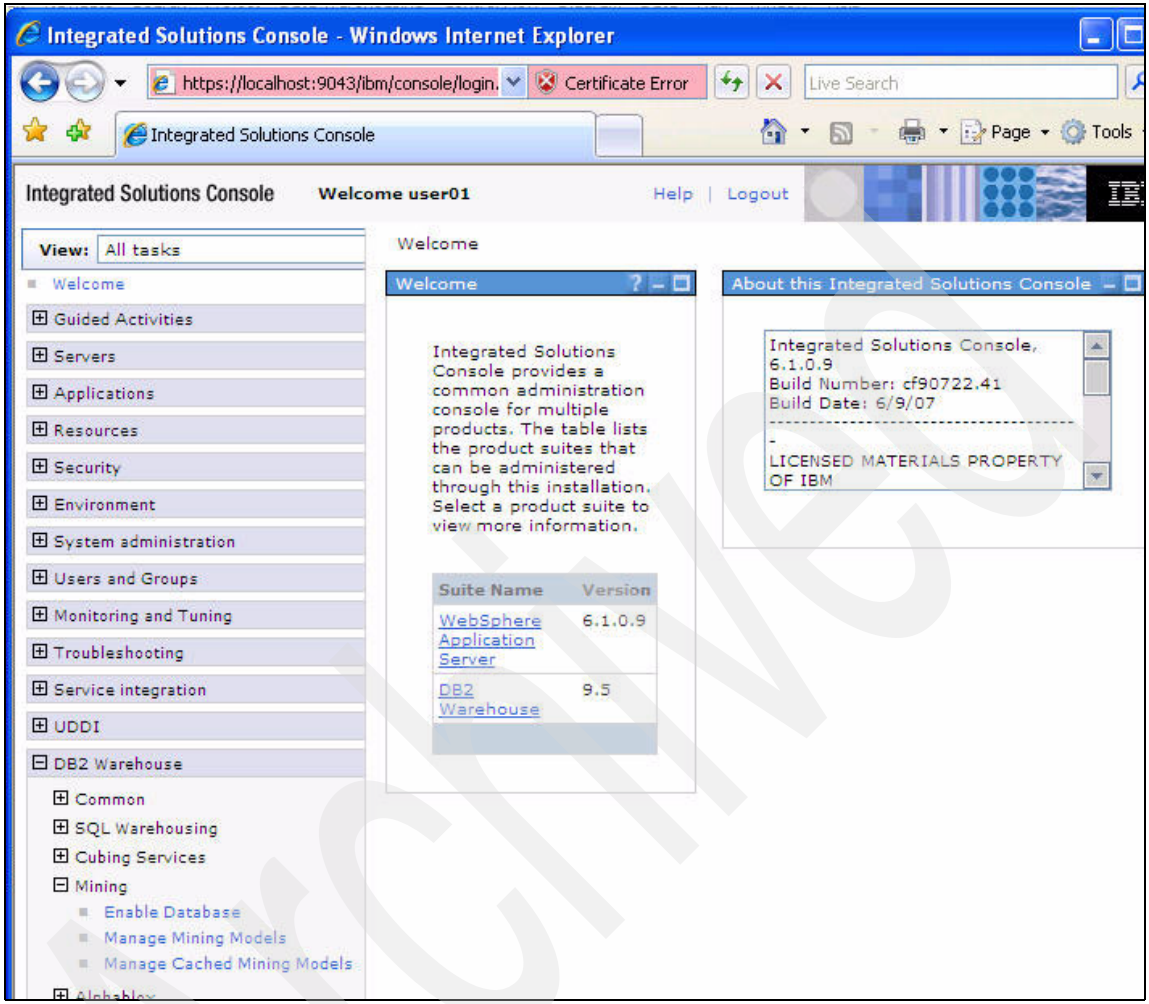


Figure 5-34 DB2 Warehouse Admin Console: main view

The *Manage Mining Models* view of the Admin Console is shown in Figure 5-35. Models can be viewed, imported, exported, deleted, or cached from this view. Except for caching, the other functions also are available in the Database Explorer view of Design Studio. To put a model into memory cache to facilitate real-time scoring, where the model persists in memory instead of being deleted from memory after a scoring process, the administrator selects the model(s) to be cached and clicks the ***Cache*** button, as shown in Figure 5-35.

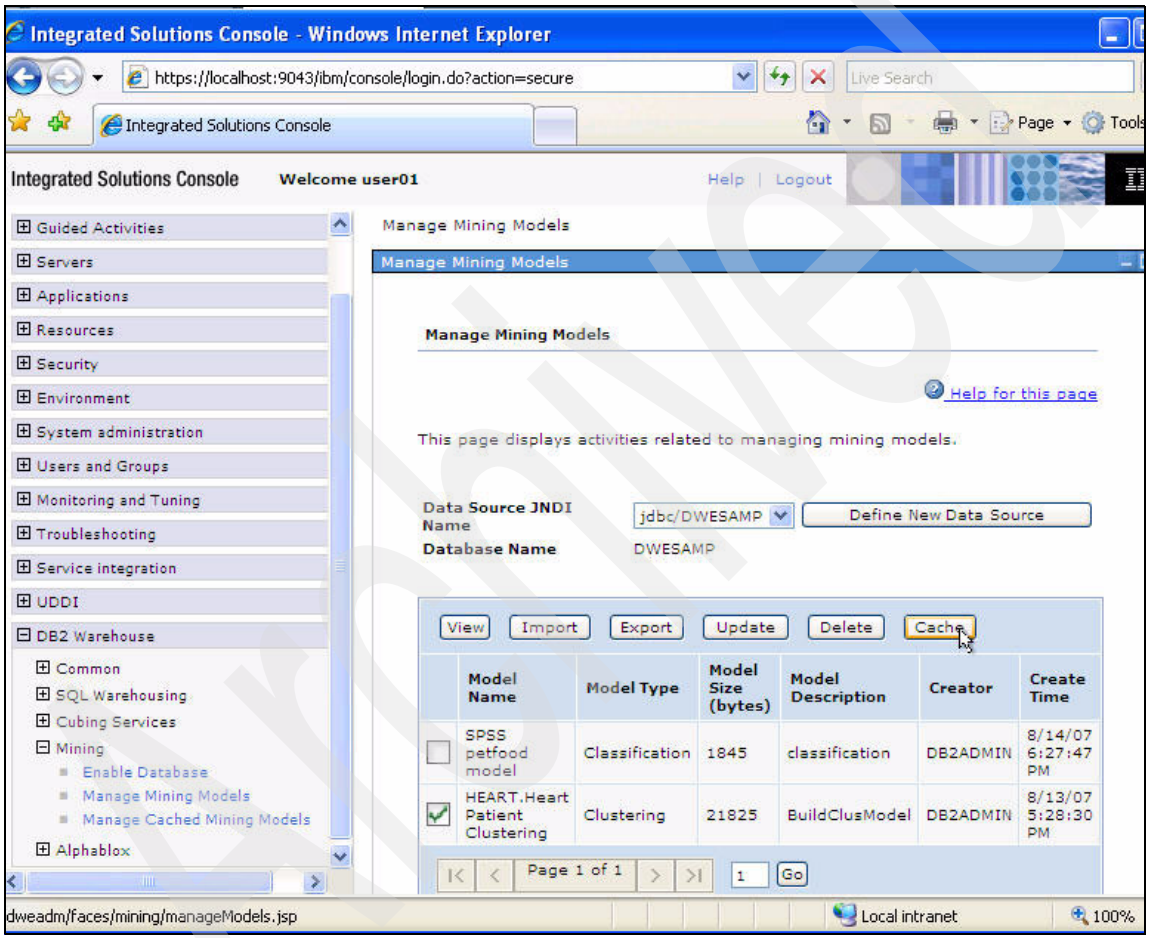


Figure 5-35 DB2 Warehouse Admin Console: managing mining models

The administrator can adjust the cache size, as shown in Figure 5-36. Clicking the **Reset Cache Size** button brings up a window in which the administrator can specify the desired cache size. In addition, models can be removed from the cache by clicking the **Remove** or **Remove All** button.

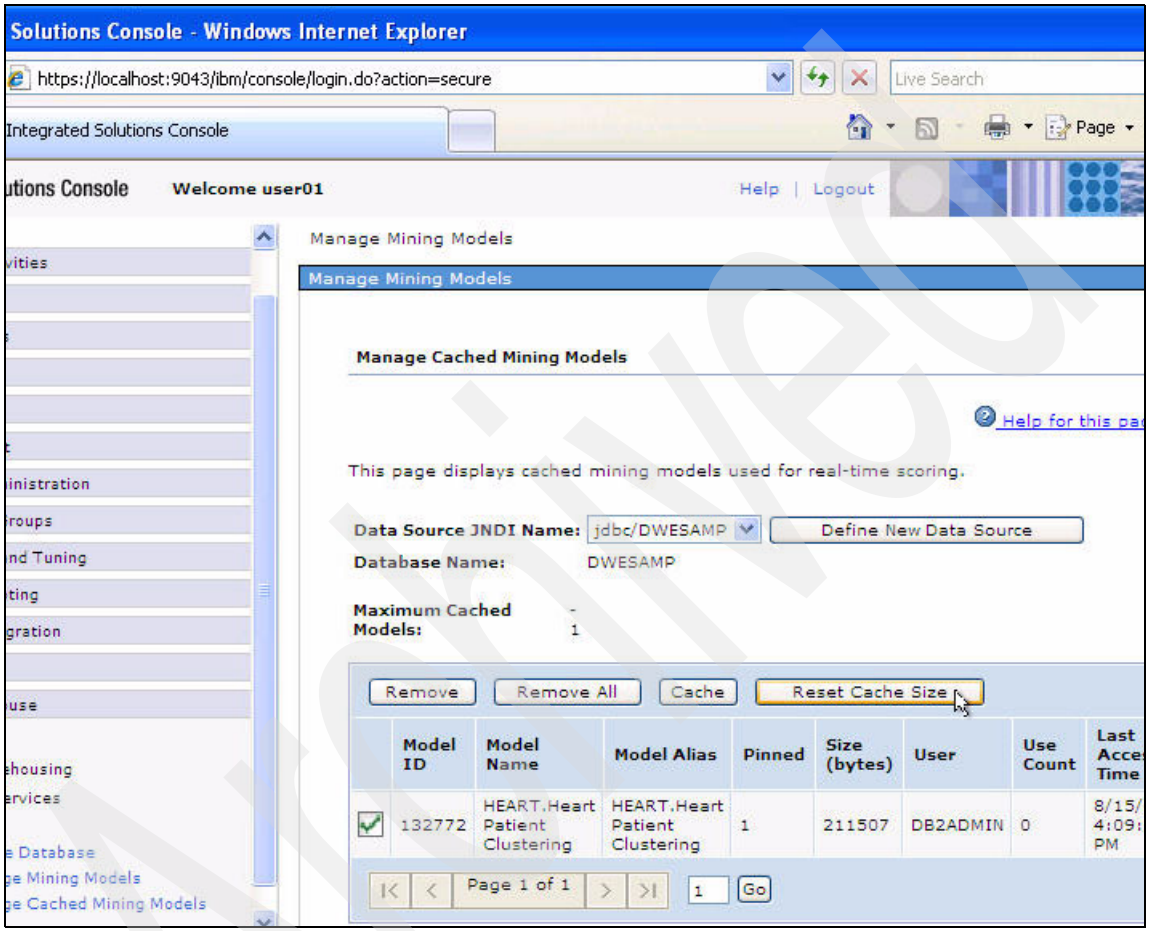


Figure 5-36 DB2 Warehouse Admin Console: managing the cache

6

Data preparation for data mining

In this chapter, we discuss the important aspects of data preparation for data mining. Data preparation is a critically important activity because it can significantly influence the results of a data mining process.

Two elements of this activity are the transformation of existing variables and the creation of new variables. These variables are called *analytical variables*, and they constitute the attributes that are used to build the data mining models. Which variables need to be created or modified depends on the business requirements. Some variables are used to build the models, while others are used just to enrich the analysis of the mining results.

This chapter includes a detailed discussion of the data preparation process for data mining, and addresses the following topics:

6.1 Data preparation as a process

Data preparation for data mining addresses an important aspect of data mining projects. Without adequate preparation of data, the return on the resources invested in mining projects is uncertain and could be disappointing. The data preparation process creates the input variables for data mining models. These variables represent the data in a context that can be explored by data mining tools, which then are used to produce insightful information to the business. No matter what the maturity, or stage of implementation, of data warehousing projects, certain levels of data preparation are still required.

Although each data mining project may have different scenarios and business goals, there is a certain pattern or structure of the data to support common business goals across different industries or business segments. For example, a data mining project to discover high-potential retail customers uses a customer segmentation model that requires a specific table structure. This data structure should contain a unique record per customer. Basically, one columns specifies the customer ID, and additional columns representing *attributes* specify customer behavior or demographic information. There are similar table structures for data mining projects in different industries, such as financial services and insurance. What differs from industry to industry is the attributes that are specific to the industry or to the problem to be solved.

Here we offer a methodology and technical details to support the data preparation process for data mining projects. Step-by-step instructions and data preparation techniques are provided in the form of case studies.

6.2 Understanding data requirements for data mining

Understanding the data requirements for data mining applications is a key factor for a successful implementation. Three major areas should be carefully considered when allocating resources for data mining projects:

1. Business requirements
2. Data requirements for data mining
3. Existing data structures and content

6.2.1 Business requirements

Understanding the business requirements is a crucial factor when performing data preparation for data mining applications. Data mining applications are business driven, but they are also data dependent. Data must be organized in such a way that the data mining process can be performed to produce the desired business results. Business requirements are the key to identifying and determining the data sources, data transformations, and new attributes and variables to be used in the data mining process.

6.2.2 Data requirements for the mining methods

Data mining methods require two different types of layout formats, depending on the method. These layouts are called *behavioral-demographic* and *transactional*.

Behavioral-demographic data layout

The term behavioral-demographic refers to the structure of the format, not to the content, which may be behavioral (as examples, total customer spending, relative percentage spending by department or product category, or shopping frequency) demographic (for example, age, gender, or income), or physical characteristics (such as temperature, speed, pressure, or store size). For more information, see *Data Preparation for Data Mining* by Pyle. In DB2 Warehouse (DB2W), the behavioral-demographic data layout is used by three mining methods: Clustering, Classification, and Regression.

The behavioral-demographic data layout consists of a single record per customer with multiple columns representing the behavioral or demographic attributes (also called *variables* in a mining model). An attribute may be categorical (for example, gender) or numeric (for example, dollars spent). The behavioral-demographic format is illustrated in the following two figures. Examples of behavioral attributes derived from transactional information are shown Figure 6-1, and examples of customer demographic attributes are shown in Figure 6-2. Note that each customer is represented by one row.

Customer ID	Total Visits	Total $ Spent	% Spent Electronics	% Spent Computers	…	% Spent Cash	…	% Spent Credit Card
…	…	…	…	…	…	…	…	…
111	44	575.00	23.8	19.32	…	5.76	…	75.63
112	48	1,322.00	13.87	22.55	…	12.80	…	53.17
113	10	180.00	1.03	43.17	..	0.00	..	100.00
…	…		…	…	…	…	…	…

Figure 6-1 Behavioral Data Layout and Attributes

Customer ID	Total Visits	Total $ Spent	Age Range	…	Gender	…	Income Range
…	…	…	…	…	…	…	…
111	44	575.00	25-30	…	Female	…	50k-60k
112	48	1,322.00	30-35	…	Female	…	130k-150k
113	10	180.00	35-40	..	Male	..	100k-120k
…	…		…	…	…	…	…

Figure 6-2 Behavioral data layout and attributes

To include both kinds of attributes in a data mining model, of course, the behavioral and demographic attributes would be combined in a single table. An example data layout including behavioral and demographic attributes is shown in Figure 6-3.

Behavioral-Demographic Format Example								
Customer ID	Total Visits	Total $ Spent	% Spent Electronics	% Spent Computers	…	Age Range	Gender	Income Range
…	…	…	…	…	…	…	…	…
111	44	575	23.8	19.32	…	25-30	Female	50k-60k
112	48	1,322	13.87	22.55	…	30-35	Female	130k-150k
113	10	180	1.03	43.17	..	35-40	Male	100k-120k
…	…		…	…	…	…	…	…

- *one record per category, such as customer or store*
- *used in Clustering, Regression, and Classification models*

Figure 6-3 Combined behavioral and demographic data layouts

Transactional data layout

The transactional data layout consists of a table definition with three columns, *Transaction*, *Item*, and *Group*.

1. Transactions are events, conditions, or states that happen at certain times. A transaction can contain a single item or multiple items (such as products, events, and conditions). It can be defined by a unique sequence identifier, by a date, or by a date or time stamp value.
2. Items are the elements of a transaction. A transaction can consist of a single item or multiple items. For example, a purchase transaction can involve one or several items (products). A part failure occurrence can consist of the failure of single or multiple components (parts). A disease episode can consist of a single symptom or multiple symptoms.

3. Transactions can also be grouped by a sequence identifier, such as Customer Number, Machine Number, or Patient Number. For example, a purchase transaction performed by a given customer can consist of one or multiple items. Purchase transactions may occur at different points of sale, such as store, internet, or some other sales channel. A customer can perform a single transaction or multiples transactions within the same day, as well several transactions over time. Figure 6-4 shows a sample transactional layout with multiple records per customer.

Customer ID	Transaction Date	Product ID
111	1/5/2007	300
111	1/5/2007	100
111	1/5/2007	130
111	4/5/2007	120
111	4/5/2007	330
112	1/2/2007	320
113	1/11/2007	100
113	1/11/2007	110
113	3/31/2007	120
113	4/1/2007	130
…	…	…

Figure 6-4 Transactional layout

This transactional layout is used by two different mining methods: Associations and Sequences.

1. The *sequences* data mining method requires all three columns in the *transactional layout*. The *item identifier* may be either character or integer, the *group identifier* may be character or numeric, and the *transaction identifier* must be either integer or date and time. The recommended format is date and time so that the calculated average number of days between transactions is meaningful when interpreting the mining results. In Figure 6-4, the *item identifier* is represented by the attribute Product ID, the *group identifier* is represented by the attribute Customer ID, and the *transaction identifier* is represented by the attribute Transaction Date.

2. The *associations* data mining method requires two columns, the *Item* and *Transaction identifiers*. The item identifier may be either character or integer, while the transaction identifier may be character, integer, or date and time. The date and time format is recommended for consistency with the required format for sequences, thus allowing the same data table to be used for both sequences and associations. In Figure 6-4, the *item identifier* is represented by the attribute Product ID and the *transaction identifier* is represented by the attribute Transaction Date.

In addition to the transactions table, optional tables for name mappings and taxonomy may be used. For example, the transactions table may contain item numbers that can be mapped to item descriptions, making the mining results much more meaningful. A taxonomy of, say, {department>class>item} can enrich the results by finding relationships across multiple levels of the product hierarchy. Figure 6-5 shows a sample table layout, which contains product name mappings and taxonomy.

Product ID	Product Subclass	Product Class	Product Department
100	Plasma TV	Televisions	Electronics
110	LCD TV large screen	Televisions	Electronics
120	LCD TV small screen	Televisions	Electronics
300	Ink Jet Printers	Printers	Computers
310	Laser B/W Printers	Printers	Computers
320	Laser Color Printers	Printers	Computers
330	Photo Printers	Printers	Computers
340	Inkjet Cartridge	Printer Accessories	Computers
…	…	…	…

Figure 6-5 Naming mappings and taxonomy for transactional format

6.2.3 Source data

Understanding of data should not be limited just to elements such as data models, tables, field names, and join keys; the data analyst should also be concerned with, and understand, data domains for elements such as categorical fields, missing values, taxonomies, outliers, data integrity, and data granularity. All these aspects should be evaluated and considered in transformation tasks for data preparation.

Categorical fields should contain valid instances that are meaningful to the business. For example, a value different from male or female in the gender field needs to be evaluated and appropriate actions taken to avoid wrong interpretations of the mining results. The same applies for fields that contain missing values. Depending on the mining algorithm being used, missing values can affect how mining models are constructed and interpreted.

Data quality assessment should be performed to analyze variables and their values, determining appropriate actions to transform variables and their values or to exclude them from the analysis. For a variable with missing values, the analyst should first determine whether “missing” actually has meaning (for example, a missing value for “number of pregnancies” when gender is male), in which case the missing values might be re-coded or the variable might be transformed into a more meaningful form. If “missing” really means that nothing is known about that variable for a particular record, then the analyst may consider whether to expend resources trying to acquire additional information about that variable (if it represents a sufficiently important effect in the model), whether to re-code the missing values with estimates generated by some scheme (as examples, average value, “hotdeck” value, and value calculated as a function of other variables), or whether it should be excluded from the model (for example, if it has a large proportion of missing values, say, 30 percent or more). Invalid or out-of-range values should be examined and re-coded if possible, or else the analyst may exclude selected records or a variable with such questionable values. Similarly, outlier values (extreme values usually with low frequency of occurrence) should be examined for validity; outliers may be simply bad data, or they may represent very interesting behavior. A variable having only a single value does not contribute any information to the model and should be excluded from the model.

The DB2 Warehouse Design Studio has a set of operators to support data transformations for data mining and also offers specific functions for data exploration (for example, univariate, bivariate, and multivariate distributions) to assist analysts in assessing data quality and gaining deep insights about the data. Details about the DB2 Warehouse Design Studio functions to support data preparation and exploration are described in the following sections of this chapter.

6.3 Data analysis and data design flow

Data analysis and data design flow are key activities in a data mining project and are executed prior to the data preparation process. The business requirements and data requirements for mining are the two major inputs for this analysis. The analyst needs to identify the gap between existing data and new data requirements. Results from this analysis include descriptions of new variables to be created, calculations and aggregations to be performed, transformations of existing data elements, and reductions in cardinality of some data elements, such as groupings of departments or product categories. In addition, all tables and relationships are identified, and the best strategy to perform the data preparation process is determined. If the data is to be sampled, then a correct sampling methodology should be used to guarantee that the sample data represents the population of customer transactions. The data analysis process is illustrated in Figure 6-6.

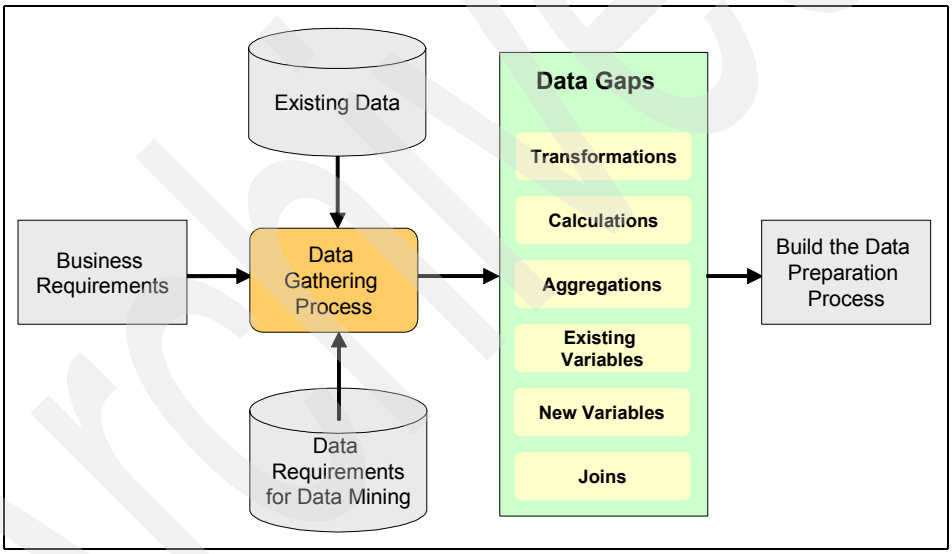


Figure 6-6 Data analysis process for data preparation

6.4 Data preparation steps

In this section, we describe the steps to perform the data preparation activity.

6.4.1 Data acquisition process

In most organizations, the data to support data mining applications is already stored in relational databases. The effort to transform the data into a suitable format that can be used for data mining depends on the characteristics of applications and databases.

Operational systems may contain only the most recent transactional data and may not be adequate for data mining applications. This data potentially requires more and complex transformations.

Typically a data mining project is a subsequent phase after the Enterprise Data Warehouse implementation (although data mining may be performed during the implementation process to help define architectural and content requirements for the warehouse). A data mining project may use a subset of the data and may be related to a business functional area such as Marketing, Finance, or Human Resources. The IBM Layered Data Architecture (LDA) for data warehousing suggests a single data repository to support different functions. These functions are defined based on the nature of access and age of the data. Three important layers in this architecture are relevant for data mining projects. They are the Operational Data Store (ODS) that contains the most recent transactional information (typically three to six months), the Enterprise Data Warehouse (EDW) that contains detailed historical and corporate information (typically three to five years), and the Data Marts that contain aggregated and business functional-oriented information (typically three to four years). These layers are depicted in Figure 6-7.

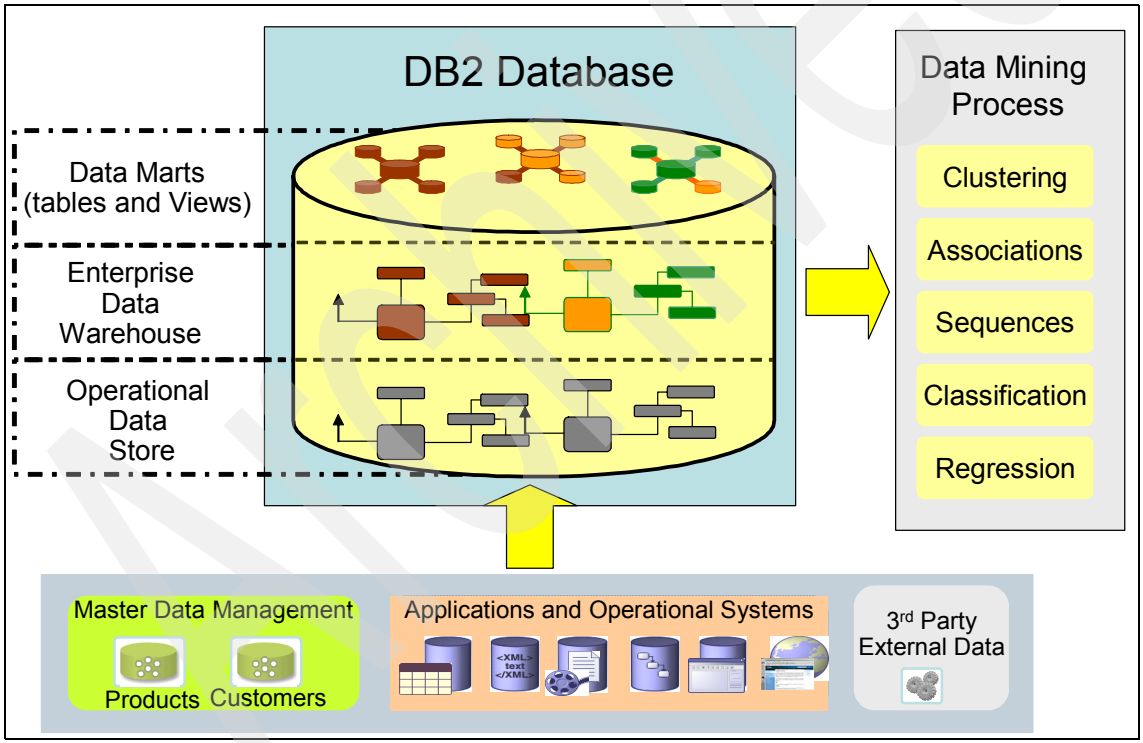


Figure 6-7 Architecture layers appropriate for data mining

Depending on the data mining application or process to be implemented, information from one or all three layers may be required. A data mart may contain aggregated customer information such as sales and returns, and it could be used to support a customer segmentation process. To discover sequences of transactions performed by a group of customers, however, it is more likely that this information is stored in the EDW or ODS.

The advantage of the LDA for data mining is that it suggests all data is to be maintained in a single database, and it avoids data movements across different servers or databases. This enables implementation of data mining projects to specific business area as well as cross-functional areas.

Because DB2W data mining algorithms are implemented within a DB2 database, the source data for the mining procedures should be stored in a relational format, and it should be available for access from the same server where the data mining product is installed. The data can be physically stored on the same server, or it can be stored in a remote server and accessed through a federated connection. Although it is possible to execute a data mining process against a remote database, there may be network performance implications to be considered.

6.4.2 Data exploration and validation

DB2 Warehouse has functionality to facilitate data exploration as part of the data preparation process. Understanding the data and validating its quality and content are essential to obtaining meaningful results from data mining analysis. In this section, we discuss four capabilities that enable the analyst to investigate data values and distributions quickly and easily through the Database Explorer:

1. Table sampling
2. Univariate distribution analysis
3. Bivariate distribution analysis
4. Multivariate distribution analysis

Table sampling

Sampling the contents of a table in the Database Explorer is illustrated in Figure 6-8. Right-clicking on the desired table brings up a pop-up menu for selecting **Data** → **Sample Contents**.

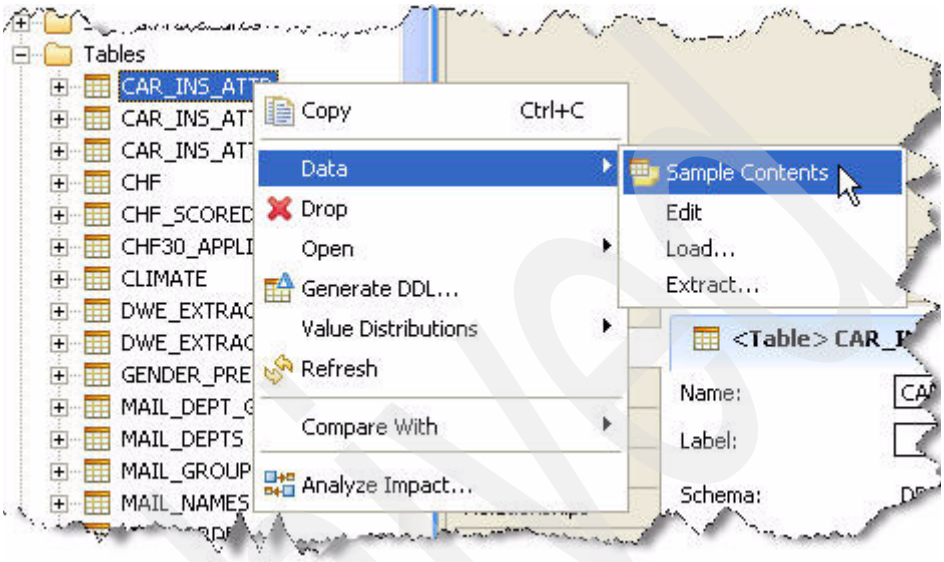
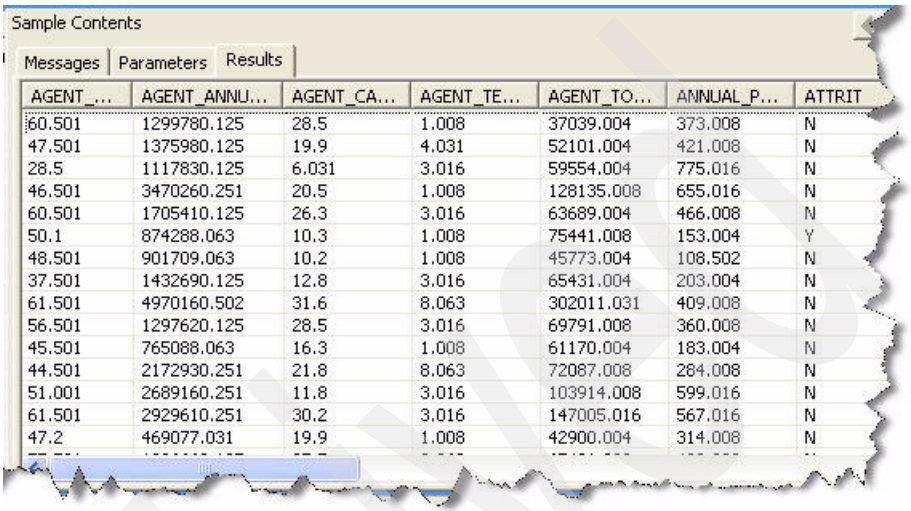


Figure 6-8 Table sampling from the Database Explorer

This selection produces a view of all the columns and a sample of the rows in the table, as illustrated in Figure 6-9. This view appears in the lower right corner of the Design Studio window. The analyst can inspect the sampled rows to get an initial feel for what the data values look like.



AGENT_...	AGENT_ANNU...	AGENT_CA...	AGENT_TE...	AGENT_TO...	ANNUAL_P...	ATTRIT
60.501	1299780.125	28.5	1.008	37039.004	373.008	N
47.501	1375980.125	19.9	4.031	52101.004	421.008	N
28.5	1117830.125	6.031	3.016	59554.004	775.016	N
46.501	3470260.251	20.5	1.008	128135.008	655.016	N
60.501	1705410.125	26.3	3.016	63689.004	466.008	N
50.1	874288.063	10.3	1.008	75441.008	153.004	Y
48.501	901709.063	10.2	1.008	45773.004	108.502	N
37.501	1432690.125	12.8	3.016	65431.004	203.004	N
61.501	4970160.502	31.6	8.063	302011.031	409.008	N
56.501	1297620.125	28.5	3.016	69791.008	360.008	N
45.501	765088.063	16.3	1.008	61170.004	183.004	N
44.501	2172930.251	21.8	8.063	72087.008	284.008	N
51.001	2689160.251	11.8	3.016	103914.008	599.016	N
61.501	2929610.251	30.2	3.016	147005.016	567.016	N
47.2	469077.031	19.9	1.008	42900.004	314.008	N

Figure 6-9 Sampled contents of a table

Preferences for distribution analysis

The number of records used in generating univariate, bivariate, and multivariate distributions is, by default, limited to a randomly selected subset of records to reduce the time to calculate the statistics for the distributions. The record limit can be adjusted by selecting **Window** → **Preferences** from the menu bar, as illustrated in Figure 6-10. This is also where record limits can be set for distribution analysis.

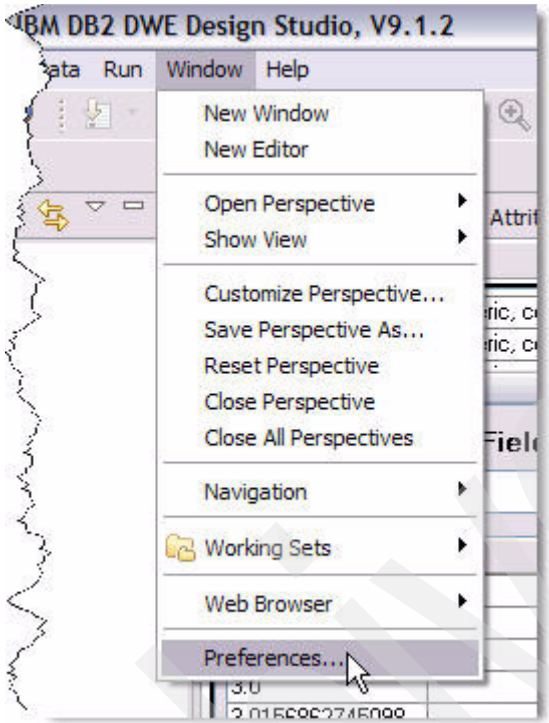


Figure 6-10 Preferences selection

For univariate and bivariate distribution analysis, the record limit is set as shown in Figure 6-11. Setting the approximate maximum to a value equal to or greater than the number of records in the table ensures that all records are used in generating the distributions.

m

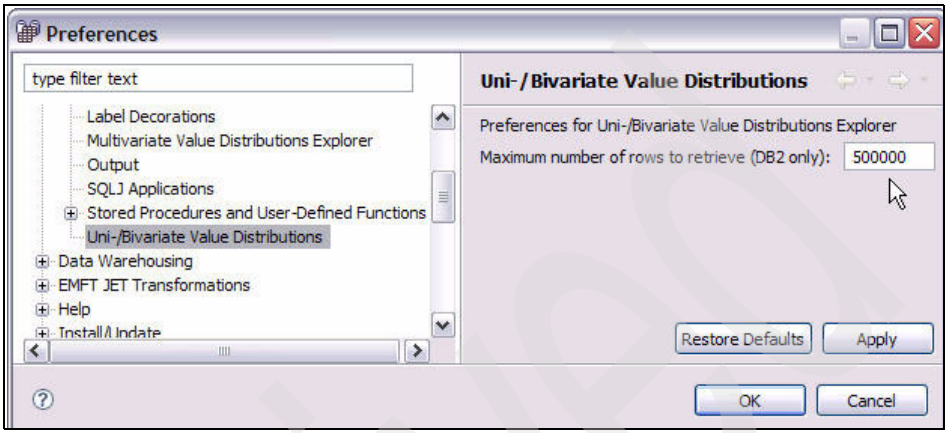


Figure 6-11 Setting the record limit for univariate and bivariate distribution analysis

For multivariate distribution analysis, the record limit is set as shown in Figure 6-12. Colors and font can also be set from this window.

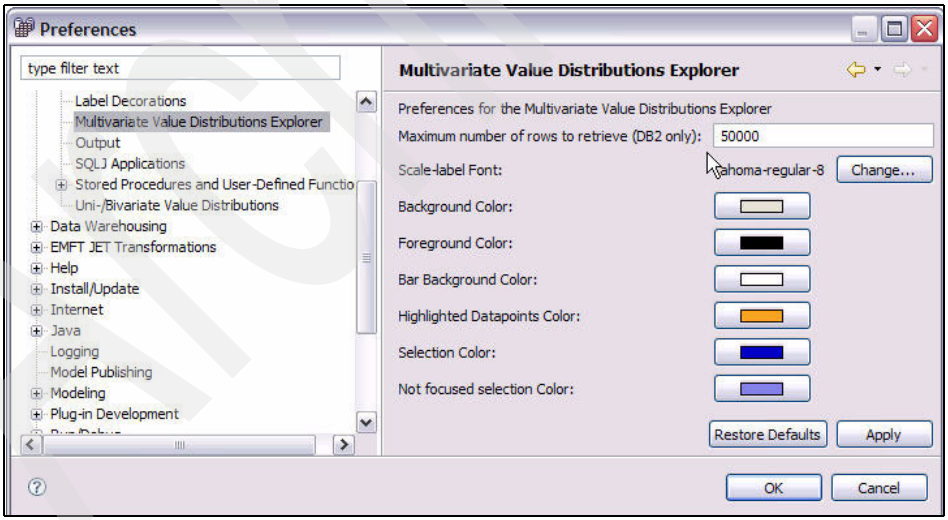


Figure 6-12 Setting the record limit, font, and colors for multivariate distribution analysis

Univariate distribution analysis

More illuminating insights about the data can be gained by analyzing the distributions of the columns in the table. The menu to select univariate, bivariate, or multivariate distributions is illustrated in Figure 6-13. Right-clicking the desired table brings up a pop-up menu for selecting **Value Distributions** and the desired view (Univariate in this case) of the data distributions as shown.

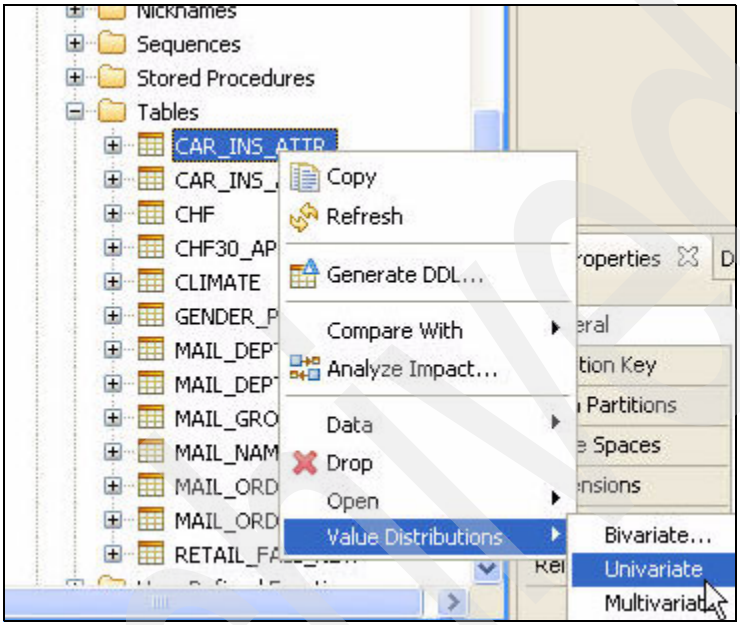


Figure 6-13 Menu for Value Distributions

Selecting **Univariate** generates a set of histograms representing the frequency distributions of the individual columns in the table. The distributions are displayed graphically, as illustrated in Figure 6-14.

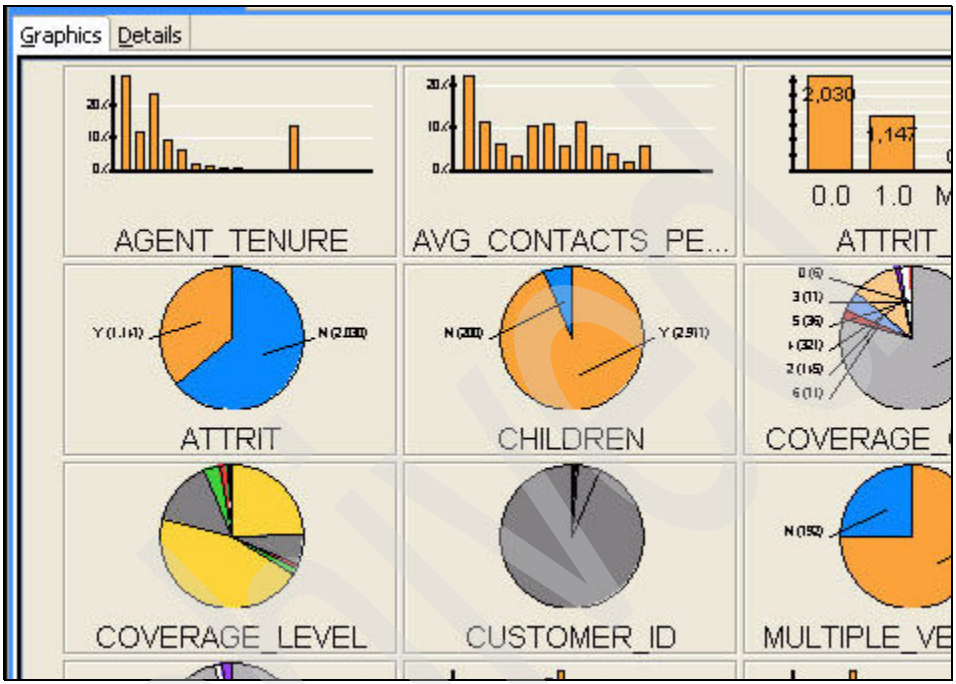
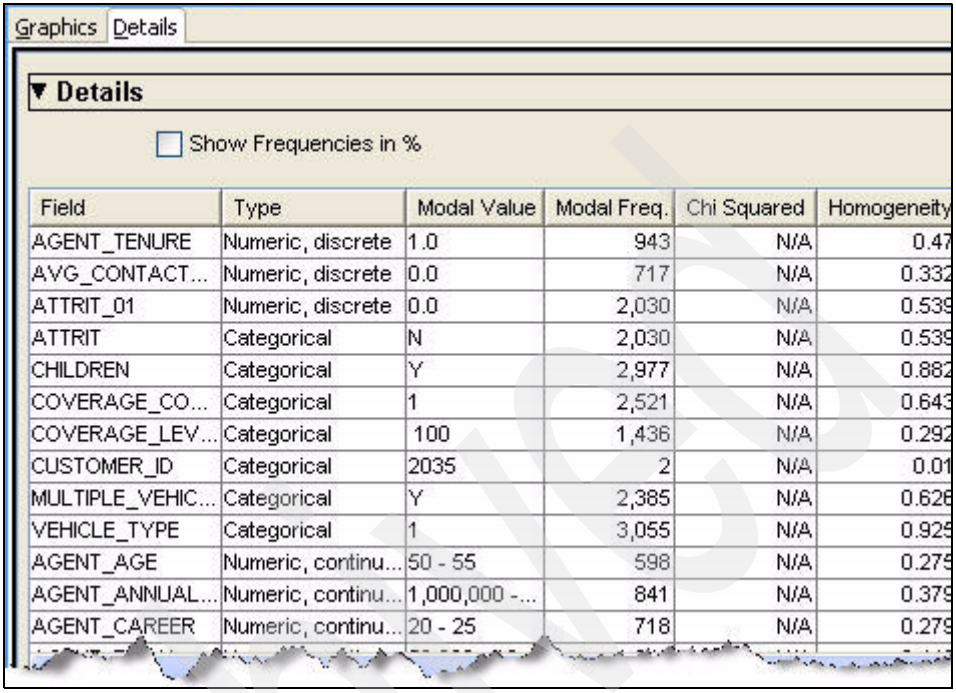


Figure 6-14 Graphical visualization of univariate distributions

The distributions are also displayed in tabular form as shown in Figure 6-15.


Graphics Details

▼ Details

Show Frequencies in %

Field	Type	Modal Value	Modal Freq.	Chi Squared	Homogeneity
AGENT_TENURE	Numeric, discrete	1.0	943	N/A	0.47
AVG_CONTACT...	Numeric, discrete	0.0	717	N/A	0.332
ATTRIT_01	Numeric, discrete	0.0	2,030	N/A	0.539
ATTRIT	Categorical	N	2,030	N/A	0.539
CHILDREN	Categorical	Y	2,977	N/A	0.882
COVERAGE_CO...	Categorical	1	2,521	N/A	0.643
COVERAGE_LEV...	Categorical	100	1,436	N/A	0.292
CUSTOMER_ID	Categorical	2035	2	N/A	0.01
MULTIPLE_VEHIC...	Categorical	Y	2,385	N/A	0.626
VEHICLE_TYPE	Categorical	1	3,055	N/A	0.925
AGENT_AGE	Numeric, continu...	50 - 55	598	N/A	0.275
AGENT_ANNUAL...	Numeric, continu...	1,000,000 -...	841	N/A	0.379
AGENT_CAREER	Numeric, continu...	20 - 25	718	N/A	0.279

Figure 6-15 Tabular display of univariate distributions

Analyzing the univariate distributions helps the analyst assess data quality and validity for an individual field (column) by revealing potential data problems such as out-of-range or invalid values that the analyst should investigate further before proceeding with any analysis.

Note: Extreme values or outliers may indicate bad data (for example, inadvertent inclusion of a few business accounts when extracting individuals' personal account records), or they may represent the most interesting behavior in the data set. Outliers should be investigated and validated before being modified or excluded from the data set.

Bivariate distribution analysis

Selecting Bivariate from the Value Distributions drop-down menu in Figure 6-13 generates a set of histograms showing how the data values of each column are distributed with respect to the values of a selected target column. Unlike univariate distributions, bivariate distributions can reveal data inconsistencies between two fields. Consider a table containing two fields (columns). The first is called GENDER with values of [M, F, U] for Male, Female, and Unknown, respectively. The second is called PREGNANCY with values of [Y, N] for Yes and No, respectively. From a data warehousing perspective, each field is considered to be “clean” because there are no invalid values within an individual field. But an inherent inconsistency is revealed when values of GENDER=M and PREGNANCY=Y occur together for a particular record.

Figure 6-16 shows the univariate distributions of the fields GENDER and PREGNANCY. For each field, all values appear valid.

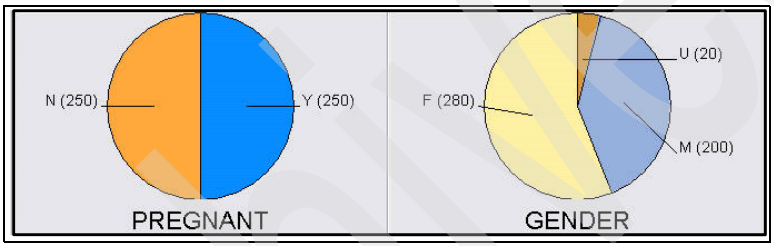


Figure 6-16 Univariate distributions for data quality problem

Figure 6-17 shows the bivariate distributions for the two fields. The bivariate distributions clearly show that some records have an inconsistent combination of values in that 10 people are coded as being both male and pregnant. The analyst must examine these records and resolve any data quality problems before proceeding with the analysis.

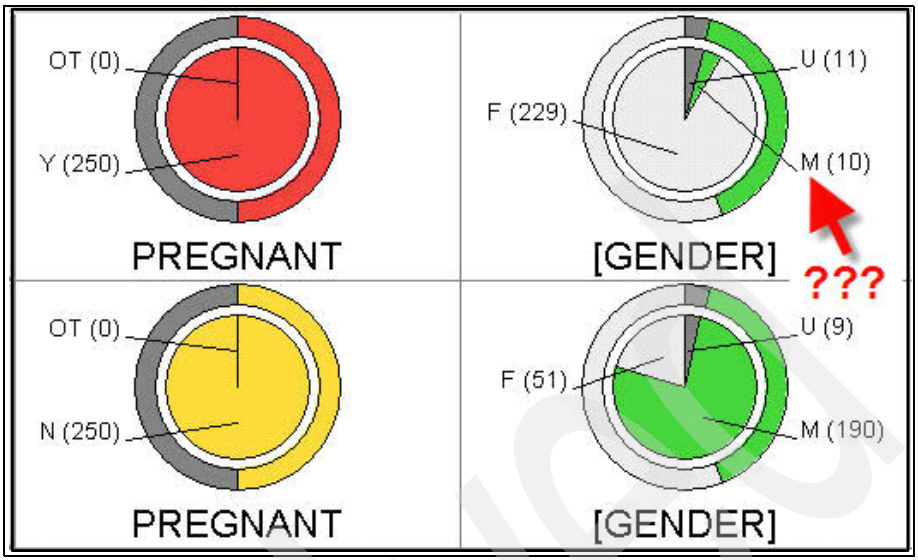


Figure 6-17 Bivariate distributions for data quality problem

Multivariate distribution analysis

Selecting **Multivariate** from the Value Distributions drop-down menu in Figure 6-13 generates a set of histograms showing how data values are distributed across all of the fields in the table, as illustrated in Figure 6-18. The analyst can select a group of records from one field's distribution and see how those records are distributed among the other fields. Multivariate distribution analysis provides insights into some of the interactions among the fields, which can help the analyst do a better job of data mining modeling and interpretation.

In the example shown in Figure 6-18, we are interested in understanding the characteristics of insurance agents serving customers who cancel their policies (attrition). Each of the 1,235 records in the data table represents a customer account. Clicking on the Y value for the field ATTRIT highlights that histogram bar and displays where those 407 customers (attriters with ATTRIT=Y) fall in the distributions of the other fields by highlighting portions of the bars in all of the histograms. In this case where the Y value of ATTRIT is selected, we see from the ATTRIT distribution that 33% of all customers attrit (407 attriters out of 1,235 customers in the sample). But for customers served by highly productive agents with annual sales of three to four million dollars (the fourth bar in the AGENT_ANNUAL_SALES histogram), the customer attrition rate is only about 24% (23 out of 96 in this subgroup). For customers served by inexperienced agents (less than 1 year of experience, represented by the first bar of the AGENT_TENURE histogram), the attrition rate is about 61% (201 out of 329 in this subgroup).

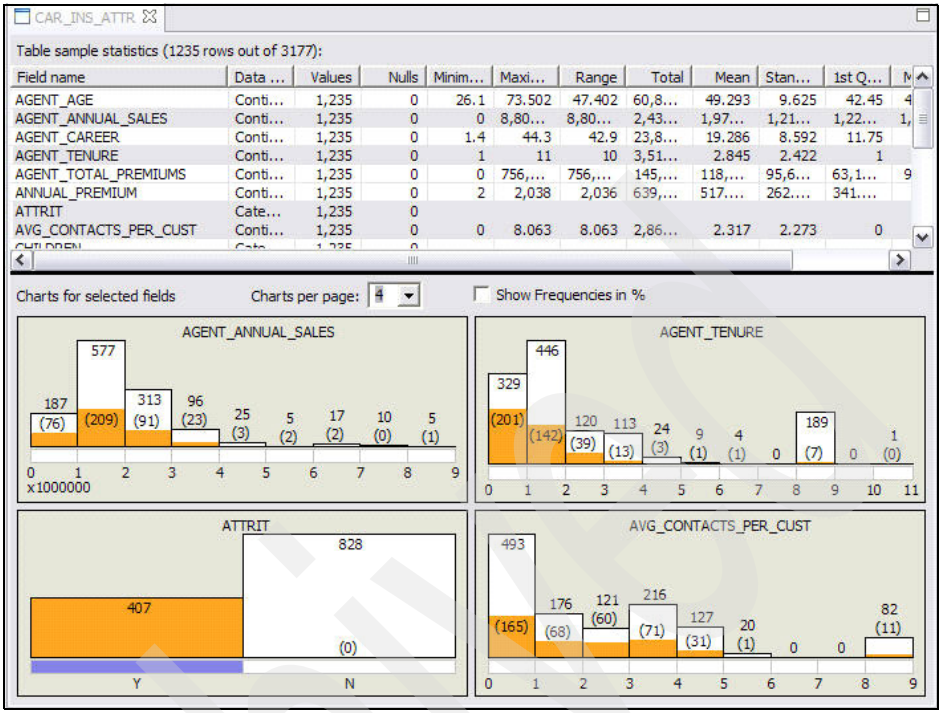


Figure 6-18 Multivariate distribution analysis for ATTRIT=Y

More detailed exploration can be done by holding down the CTRL key and selecting multiple histogram bars. As illustrated in Figure 6-19, clicking on the **Y** value for ATTRIT and the **[0-1]** value for AGENT_TENURE shows how the records having both these values are distributed across all fields.

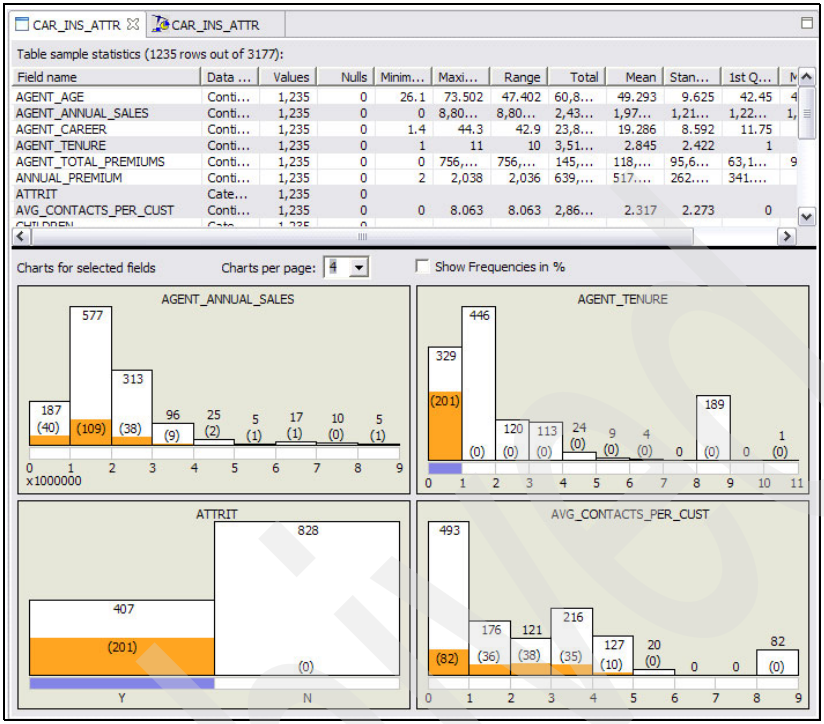


Figure 6-19 Multivariate distribution analysis

6.4.3 Transforming the data

Data transformation is a very important step in the data preparation process. How transforms are performed can significantly influence the mining results. If existing datasets were created for generic use, such as reporting or transactional applications, additional transformations very likely will be required for data mining modeling. Low-quality or inappropriately-expressed data will produce low-quality mining models that will be of little or no value in solving a business problem. Time and resources spent in data preparation for data mining is certainly a good investment and will lead to better data mining models.

There are two major reasons to perform data transformations. These are to:

1. Create new variables representing attributes or characteristics (also called analytical variables)
2. Transform data to enhance data quality or usability in mining models

Creation of new variables

For data mining purposes, the data available in the database commonly consists of just basic, raw columns in database tables. Even if the data to be used in the data mining project is sourced from a existing data warehouse, new variables still need to be created to represent the data appropriately for mining. In general, a data mining project has very specific business goals, and the mining variables need to be to expressed in the way most relevant to these business goals. Data marts tailored to specific data mining needs can greatly reduce the total time to complete a data mining project by reducing the data preparation time.

New variables are generated from existing (raw) variables or columns in database tables. Transformations (typically based on mathematical formulas involving logic or other variables) are applied to create the new variables. These new variables can be business-specific expressions, or they could be logical expressions such as flags (yes/no). To generate business-specific variables, the analyst must have a very good understanding of the business problem to be solved as well as a very good understanding of the existing data. These business-specific variables represent meaningful information, such as a certain ratio, a relative value, or a measurement to represent magnitude, size, quantity, or amount. These variables have specific meaning within the context of the business and thus can provide insights in the interpretation of the mining results.

The following are examples of variables commonly used in marketing applications. They are easily calculated from existing variables:

- Average dollar amount spent over a specific period of time
- Average number of transactions over a specific period of time
- Average number of items per transactions over a specific period of time
- Count of distinct items purchased over a specific period of time
- Count of distinct departments, stores, or other groups in which transactions are performed over a specific period of time
- Number of total transactions performed over a specific period of time
- Total dollar spent over a specific period of time
- Total quantity spent over a specific period of time
- Total dollar returned over a specific period of time
- Total quantity returned over a specific period of time
- Percent spent on specific departments, products, product categories, or other groups over a specific period of time
- Percent spent on each week of the year, month of the year, day of the week, and so forth

- Percent of total spent in a specific period of time relative to the available credit in the same period
- Highest price paid for a certain product over a specific period of time
- Percent of discount over a specific period of time
- Percent of spending by sales channel (for example, internet or in-store)
- Ratio of price paid for most expensive item to least expensive item
- Ratio of total debit to available line of credit

In addition to these examples, there are many other variables that could be created for use in data mining models. The important considerations are that they need to be meaningful in the context of the business problem, and they need to contribute to improving model quality and relevance.

Data preparation and transformation examples

In this section, we describe the process of transforming data for data mining. The amount of effort required for this activity can vary considerably since the number of transformations required depends on the business objectives of the data mining project and the existing information available at the start. To illustrate this process, we use examples of table layouts that contain raw data not in the necessary form for data mining. The goal of this section is to demonstrate the major steps required to transform the data into a format that can be used in a data mining process.

The table in Figure 6-20 contains a sample of transactional source data that is typically available in the retail business. This data could vary in content and format, so this is just a sample data layout to illustrate how to apply transformations and generate a *customer behavior* table and *customer transaction* table for a data mining process.

Customer ID	Trans Date	Product ID	Quantity	Sales Amount	Payment Type
111	01/05/2007	300	1	250.20	CREDIT CARD
111	01/05/2007	100	2	30.09	CREDIT CARD
111	01/05/2007	130	4	60.50	CREDIT CARD
111	04/05/2007	120	2	32.23	CASH
111	04/05/2007	330	1	110.00	CASH
112	01/02/2007	320	1	550.00	CREDIT CARD
113	01/11/2007	100	1	8.49	CASH
113	01/11/2007	110	2	20.00	CASH
113	03/31/2007	120	1	15.34	DEBIT CARD
113	04/01/2007	130	2	40.00	CASH
114	03/15/2007	120	1	50.00	CREDIT CARD
114	01/05/2007	320	1	500.00	CREDIT CARD
…	…	…	…	…	…

Figure 6-20 Sample of typical transaction sales data: source data for data mining

The table in Figure 6-21 shows sample data to be used as taxonomy in the data mining models. The taxonomy represents a hierarchy of products and higher-level groupings.

Product ID	Product Name	Product Subclass	Product Class	Product Department
100	42" Flat-Panel LCD HDTV	LCD Flat Panel	Televisions	Electronics
110	52" Flat-Panel LCD HDTV	LCD Flat Panel	Televisions	Electronics
120	46" Flat-Panel Plasma HDTV	Plasma Flat Panel	Televisions	Electronics
130	52" Flat-Panel Plasma HDTV	Plasma Flat Panel	Televisions	Electronics
300	12PPM Color, Inkjet Printer	Ink Jet Printers	Printers	Computers
310	35PPM B/W, Laser Printer	Laser B/W Printers	Printers	Computers
320	30PPM Col. Laser Printer	Laser Color Printers	Printers	Computers
330	1200 DPI Portable Photo Printer - Black/Silver	Photo Printers	Printers	Computers
…	…	…	…	…

Figure 6-21 Sample of typical product taxonomy data

An example of a behavioral data layout, which could be created using the information sourced from the source data depicted in Figure 6-20 and Figure 6-21, is shown in Figure 6-22. The variables (columns) in this table are, of course, merely examples and could vary depending on the business objectives. The goal here is to demonstrate how to transform the data in Figure 6-20 and Figure 6-21 into the format depicted in Figure 6-22.

Cust ID	Total Visits	Total Spent	Total Items	PCT Spent Cash	PCT Spent Credit Card	PCT Spent Debit Card	PCT Spent Jan	PCT Spent Feb	PCT Spent Mar	PCT Spent Apr	PCT Spent Electronics	PCT Spent Computers	PCT Spent ...
111	4	483.02	5	29.45	70.55	0	70.55	0	0	29.45	74.57	25.43	...
112	1	550.00	1	0	100	0	100	0	0	0	100	0	...
113	4	83.83	4	81.71	0	18.29	33.99	0	18.29	47.72	0	100	...
114	2	550	2	0	100		9.10	0	90.9	0	90.9	9.10	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 6-22 Customer purchase behavior - sample data layout

The data in Figure 6-22 represents one record per customer. The variables have the following meanings:

Cust ID Unique customer identification number

Total Visits Total customer visits from January to April

Total Spent Total customer spending from January to April

Total Items Total items customer purchased from January to April

PCT Spent Cash Percent customer spent in cash

PCT Spent Credit Card Percent customer spent with credit card

PCT Spent Debit Card Percent customer spent with debit card

PCT Spent Jan Percent customer spent in January

PCT Spent Feb Percent customer spent in February

PCT Spent Mar Percent customer spent in March

PCT Spent Apr Percent customer spent in April

PCT Spent Electronics

Percent customer spent in the electronics department from January to April

PCT Spent Computers

Percent customer spent in the computers department from January to April

PCT Spent ... Percent Customer spent in each department from January to April

Examples of transformations using Design Studio

In this section, we describe the steps to perform the transformations that populate the sample table in Figure 6-22. A DB2 Warehouse Design Studio Data Flow, used to build the *Customer Purchase Behavior* table, is shown in Example 6-23. This example uses the *unpivot operator* to create the new data mining variables. Alternatively, the data flow could use *case statement* transformations instead of the *unpivot* operator and obtain the same results.

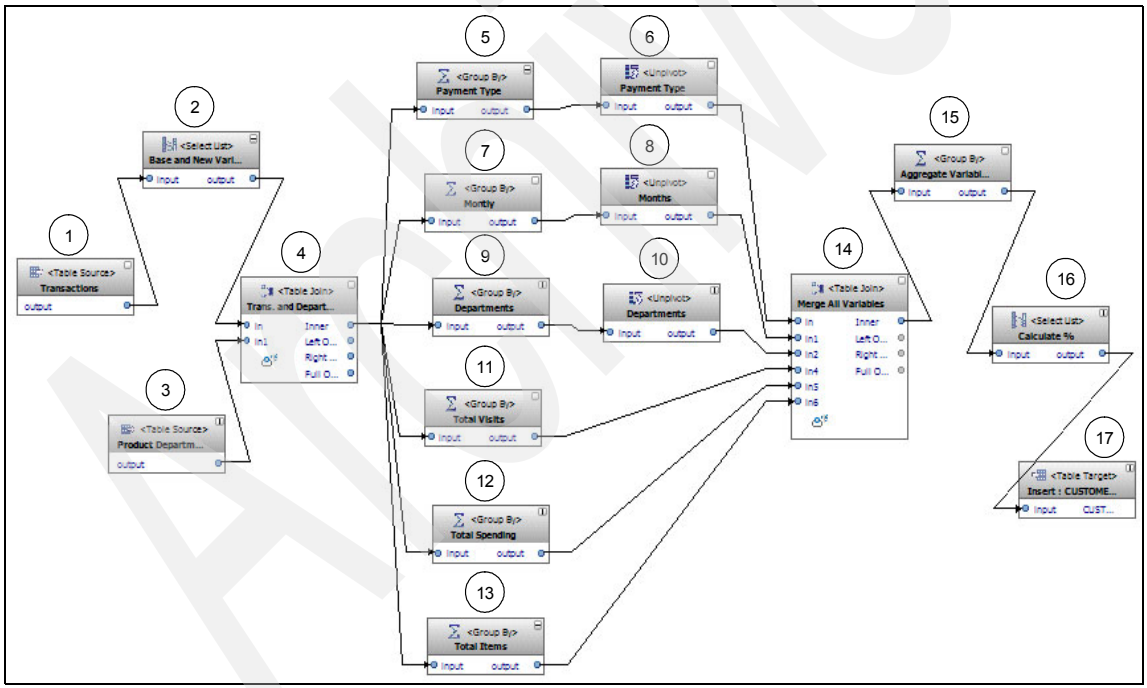


Figure 6-23 Sample Behavioral Data Flow - Populate Customer Behavior Table

The following are descriptions of the data flow operations in Example 6-23 on page 125:

- Operator 1: Source Transaction Table.
- Operator 2: Select all required information from transaction table. This operator has one transformation to create a new column called MONTH_NAME. The month_name column is used to calculate the percentage spent by month for each customer. See Figure 6-24.

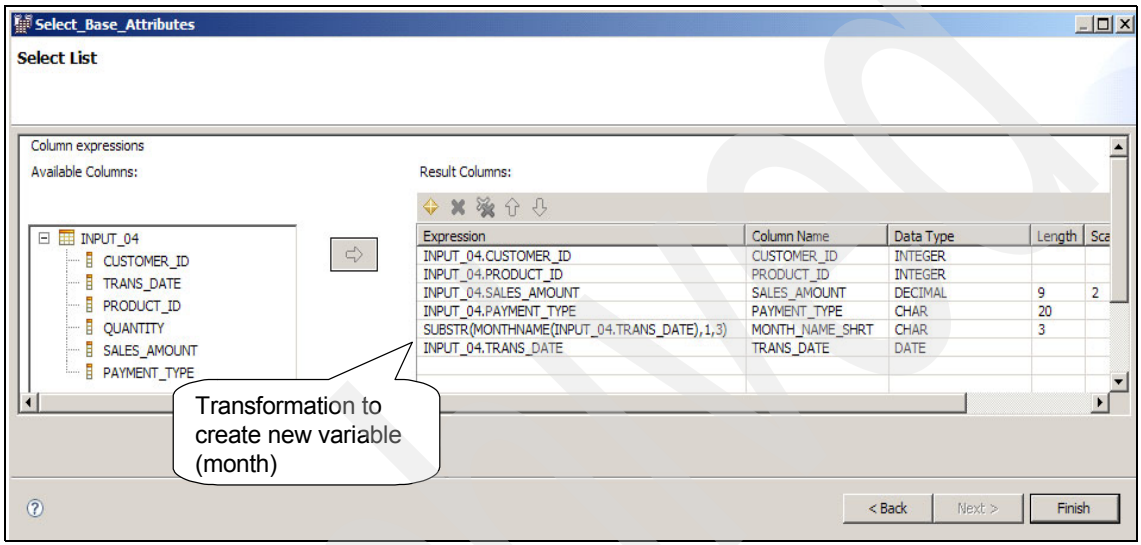


Figure 6-24 Operator 2: Select transaction information and create Month_Name variable

- Operator 3: Select product group information used to calculate percentage of spending by product group for each customer.
- Operator 4: Join definition between transactions and product tables.

- Operator 5: Use the Group By operator to aggregate sales amount for each customer by different payment types. See Figure 6-25.

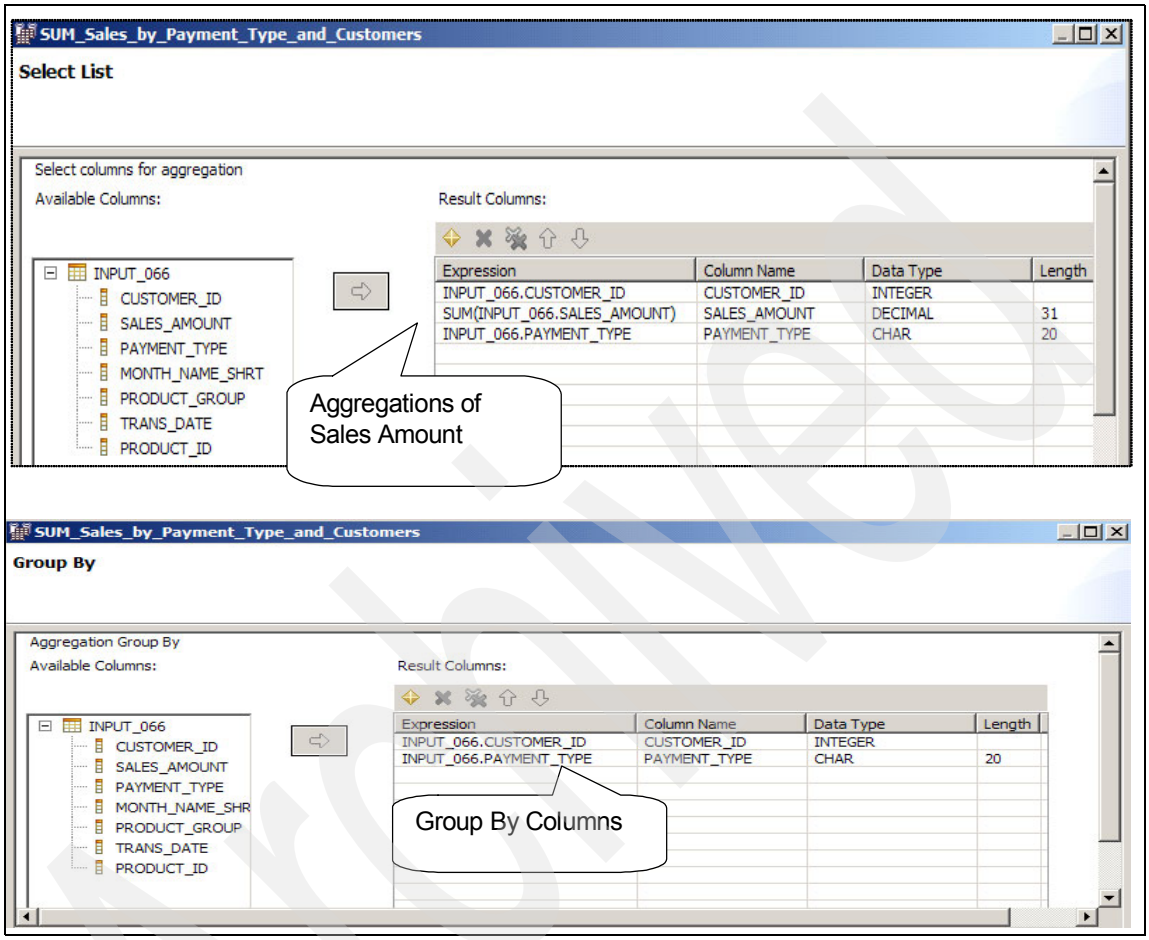


Figure 6-25 Operator 5: Aggregation of sales amount

- Operator 6: Use the unpivot operator to calculate the total of spending by each customer using different payment types. As result of this operator, three additional variables are generated: TOTAL SPENDING CASH, TOTAL SPENDING CREDIT CARD, and TOTAL SPENDING DEBIT CARD. See Figure 6-26.

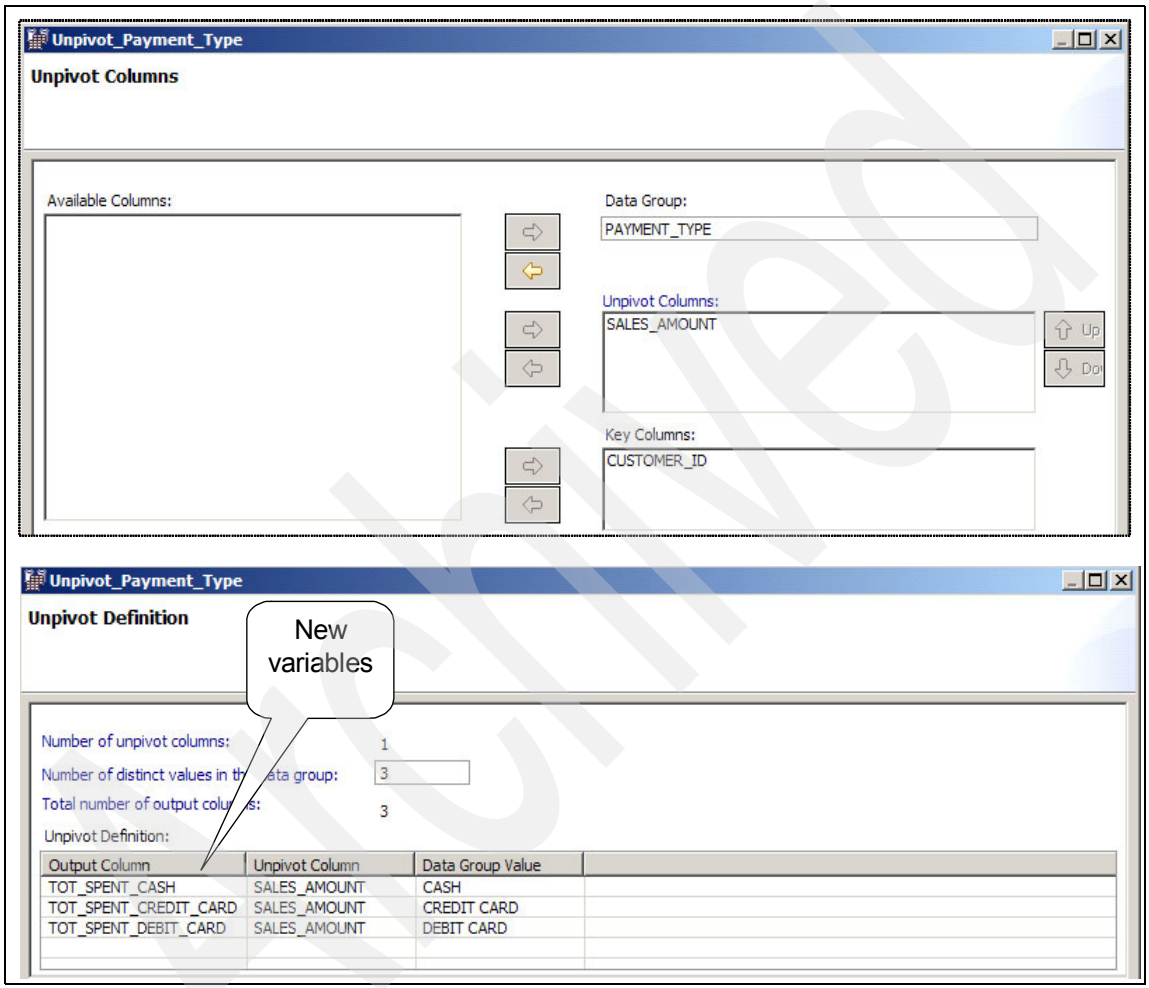


Figure 6-26 Operator-6: Unpivot payment type

- Operator 7: Use the Group By operator to aggregate the sales amount for each customer by different months.

- Operator 8: Use the unpivot operator to calculate the total spending by each customer in each month. As result of this operator, four additional variables are generated: TOTAL SPENDING JAN, TOTAL SPENDING FEB, TOTAL SPENDING MAR, and TOTAL SPENDING APR.
- Operator 9. Use the Group By operator to aggregate the sales amount for each customer by different product groups.
- Operator 10: Use the unpivot operator to calculate the total spending by each customer on each product department. As result of this operator, additional variables are generated, such as TOTAL SPENDING ELECTRONICS and TOTAL SPENDING COMPUTERS
- Operator 11: Use the Group By operator to calculate the total number of visits for each customer. As result of this operator, one additional variable is generated: TOTAL VISITS. See Figure 6-27.

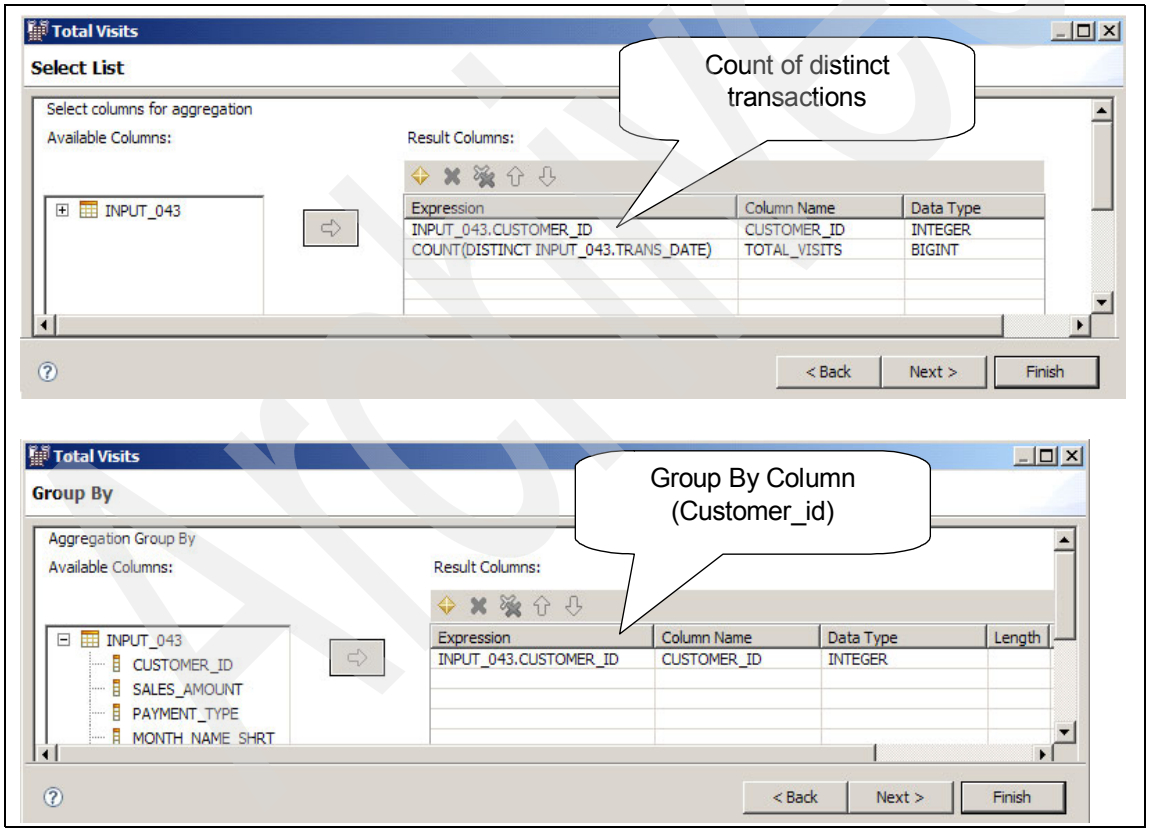


Figure 6-27 Operator 11: Create Total_Visits variable

- Operator 12: Use the Group By operator to calculate the total spent for each customer. As result of this operator, one additional variable is generated: TOTAL SPENT.
- Operator 13: Use the Group By operator to calculate the total items for each customer. As result of this operator, one additional variable is generated: TOTAL ITEMS. See Figure 6-28.

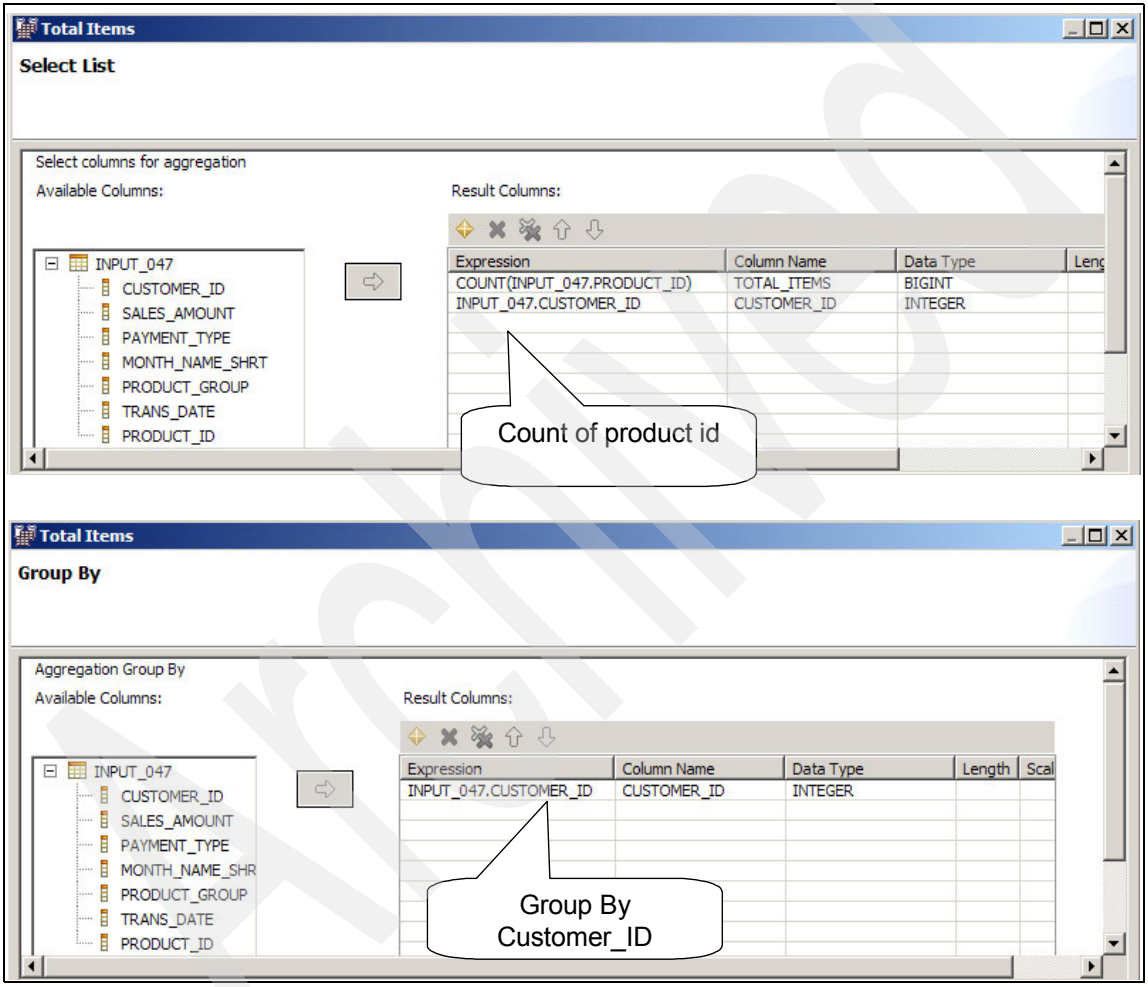


Figure 6-28 Operator 13: Create Total Items variable

- Operator 14: Use the Table Join operator to join all existing and new variables into a single table.

- Operator 15: Use the Group By operator to aggregate all new and existing variables into a single table. This operation results in a single record per customer. See Figure 6-29.

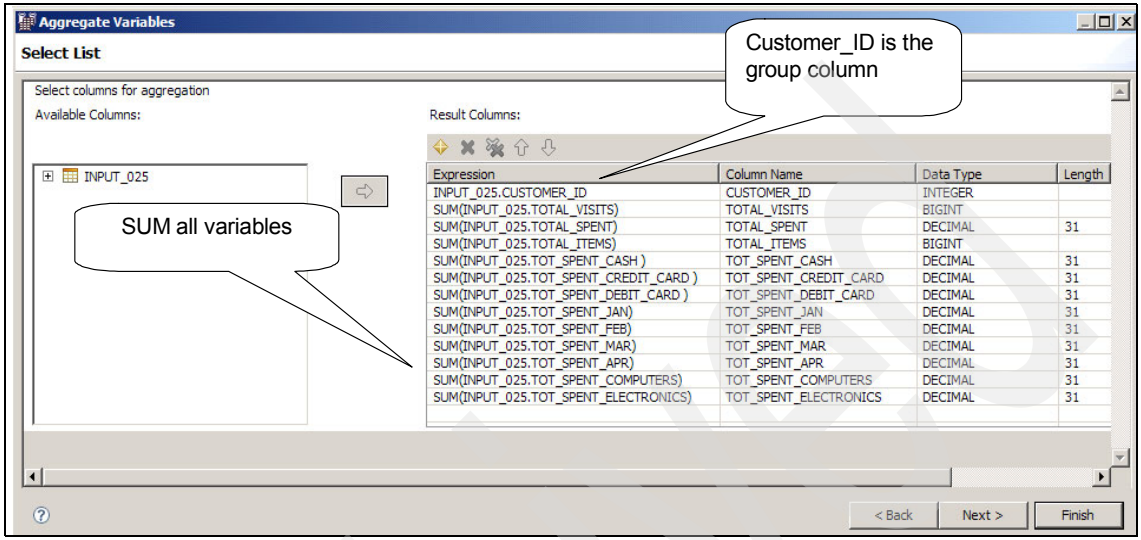


Figure 6-29 Operator 15: Aggregate all variables for each customer

- Operator 16: Use the select list operator to calculate the percentages for all variables that should contain percent ratios. See Figure 6-30.

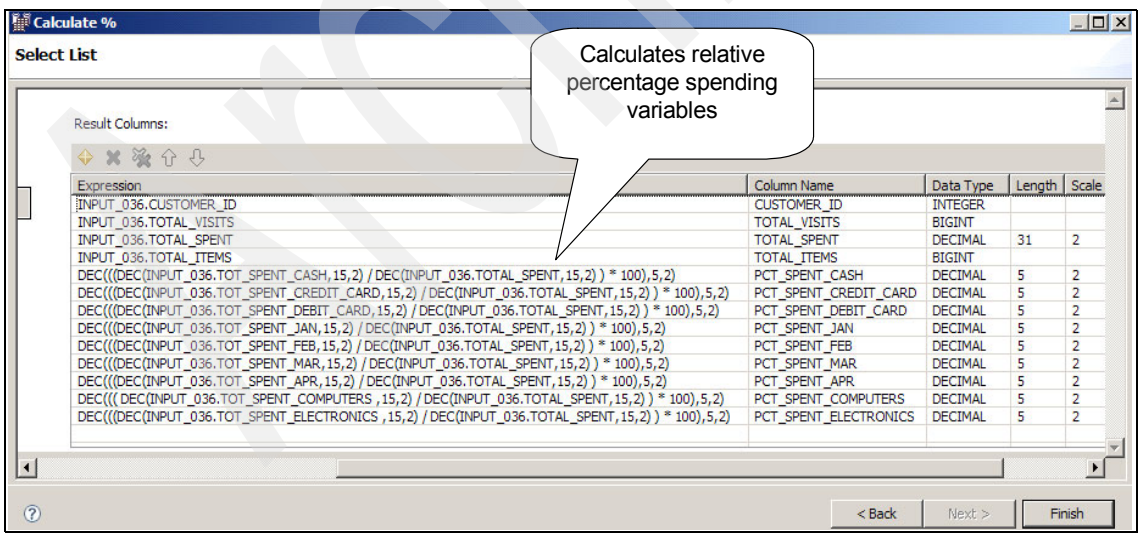


Figure 6-30 Operator 16: Calculation of percentages variables

- Operator 17: Use the Insert operator to insert all rows into the CUSTOMER_PURCHASE_BEHAVIOR table.The result set table is one record per customer.

The DDL for the target customer behavior table used with operator 17 is shown in Figure 6-31. This table contains the same columns definitions that corresponds to all columns shown in Figure 6-30. With operator 17, all the source transformed columns are mapped to the target customer behavior table.

The target customer behavior table contains the customer key and behavioral attributes. Note that some of the variables are expressed in total (TOTAL_ITEMS, TOTAL_VISITS, and TOTAL_SPENDING), while some other variables are expressed in percentages (for example, PCT_SPENT_COMPUTERS = percent each customer spent in the computer department).

```
CREATE TABLE ITSO.CUSTOMER_PURCHASE_BEHAVIOR  (

        CUSTOMER_ID                 INTEGER NOT NULL,
        TOTAL_VISITS                BIGINT ,
        TOTAL_SPENT                 DECIMAL(31,2) ,
        TOTAL_ITEMS                 BIGINT ,
        PCT_SPENT_CASH              DECIMAL(5,2) ,
        PCT_SPENT_CREDIT_CARD       DECIMAL(5,2) ,
        PCT_SPENT_DEBIT_CARD        DECIMAL(5,2) ,
        PCT_SPENT_JAN               DECIMAL(5,2) ,
        PCT_SPENT_FEB               DECIMAL(5,2) ,
        PCT_SPENT_MAR               DECIMAL(5,2) ,
        PCT_SPENT_APR               DECIMAL(5,2),
        PCT_SPENT_ELECTRONICS       DECIMAL(5,2) ,
        PCT_SPENT_COMPUTERS         DECIMAL(5,2))
 IN USERSPACE1 ;


ALTER TABLE ITSO.CUSTOMER_PURCHASE_BEHAVIOR
        ADD CONSTRAINT 'PK_CUSTOMER'  PRIMARY KEY (CUSTOMER_ID);
```

Figure 6-31 DDL for customer purchase behavior table

Examples of transformations using SQL language

The recommended data preparation approach is to use Design Studio to perform the transformations. Design Studio stores metadata and transformations in a repository to facilitate future maintenance. Additionally, data flows can be integrated with mining flows into a control flow that facilitates the execution of the whole data mining process.

If Design Studio is not the tool of choice for data preparation, then any Extract, Transform and Load (ETL) tool or SQL scripts can be used in the data preparation process for data mining. In this section, we describe an alternative way to perform data preparation using SQL scripts.

This alternative process is demonstrated using the same sample data layouts shown in Figure 6-20, Figure 6-21, and Example 6-23 on page 125. The ultimate goal of this process is to create transformations to populate the target customer behavior table shown in Figure 6-31. The two source tables layouts used in the process are shown in Figure 6-32.

```
CREATE TABLE ITSO.TRANSACTIONS  (
    CUSTOMER_ID         INTEGER         NOT NULL ,
    TRANS_DATE          DATE            NOT NULL ,
    PRODUCT_ID          INTEGER         NOT NULL ,
    QUANTITY            INTEGER         NOT NULL ,
    SALES_AMOUNT        DECIMAL(9,2)    NOT NULL ,
    PAYMENT_TYPE        CHAR(20)        NOT NULL )
IN USERSPACE1 ;



CREATE TABLE ITSO.PRODUCTS (
    PRODUCT_ID              INTEGER  PRIMARY KEY NOT NULL ,
    PRODUCT_NAME            VARCHAR(50) ,
    PRODUCT_SUBCLASS         VARCHAR(40) ,
    PRODUCT_CLASS           VARCHAR(40),
    PRODUCT_DEPARTMENT  VARCHAR(40)
 IN USERSPACE1 ;
```

Figure 6-32 Products and transactions: sample source table layouts

The source tables in Figure 6-32 contain sales and product information. The source table TRANSACTIONS contains detailed sales information with one record per customer per transaction (trans_date). This table also contains payment type information (Credit Card, Cash, and Debit Card). The source table PRODUCTS contains product characteristics information (name, class, subclass, and department).

The SQL language allows the use of different approaches to achieve the same results. Here we describe one of the possible approaches that has been used in a particular customer scenario, with reasonable performance. In this scenario, the behavioral customer attributes are expressed in percentages to represent behaviors rather than merely “small-medium-large” spending levels. To express the relative behaviors, it is necessary to calculate the total customer spending in each department, the amount spent per payment type, and the amount spent each month, then calculate the percentage spent by department.

The first step in these calculations is to allocate the amount that each customer spent in each month, department, and type of payment. This requires a join between the products and transactions tables. This is illustrated in Figure 6-33. Note that the group by clause of the SQL statement still has multiple columns (CUSTOMER_ID, TRANS_DATE, PRODUCT_TYPE, and PAYMENT_TYPE), which will generate multiple records per customer. To populate the final target table, there should be only one record per customer.

```
CREATE VIEW ITSO.ALLOCATE_SPENDING_VIEW AS

SELECT

T1.CUSTOMER_ID                          AS CUSTOMER_ID,
COUNT(DISTINCT T1.TRANS_DATE)           AS TOTAL_VISITS,
SUM(T1.SALES_AMOUNT)                    AS TOTAL_SPENT,
COUNT(T2.PRODUCT_ID)                    AS TOTAL_ITEMS,

CASE WHEN T1.PAYMENT_TYPE = 'CASH'          THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END  AS TOT_SPENT_CASH,
CASE WHEN T1.PAYMENT_TYPE = 'CREDIT CARD'   THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END  AS TOT_SPENT_CREDIT_CARD,
CASE WHEN T1.PAYMENT_TYPE = 'DEBIT CARD'    THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END  AS TOT_SPENT_DEBIT_CARD,

CASE WHEN SUBSTR(MONTHNAME(T1.TRANS_DATE),1,3) = 'JAN' THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END AS TOT_SPENT_JAN,
CASE WHEN SUBSTR(MONTHNAME(T1.TRANS_DATE),1,3) = 'FEB' THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END AS TOT_SPENT_FEB,
CASE WHEN SUBSTR(MONTHNAME(T1.TRANS_DATE),1,3) = 'MAR' THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END AS TOT_SPENT_MAR,
CASE WHEN SUBSTR(MONTHNAME(T1.TRANS_DATE),1,3) = 'APR' THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END AS TOT_SPENT_APR,

CASE WHEN T2.PRODUCT_GROUP = 'ELECTRONICS' THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END  AS TOT_SPENT_ELECTRONICS,
CASE WHEN T2.PRODUCT_GROUP = 'COMPUTERS'   THEN SUM(SALES_AMOUNT) ELSE SUM(0)   END  AS TOT_SPENT_COMPUTERS


 FROM  ITSO.TRANSACTIONS T1,
       ITSO.PRODUCTS     T2

WHERE T1.PRODUCT_ID = T2.PRODUCT_ID

 GROUP BY    T1.CUSTOMER_ID,
             T1.TRANS_DATE,
             T1.PAYMENT_TYPE,
             T2.PRODUCT_GROUP;
```

Figure 6-33 Allocate customer spending: sample SQL transformation

After data has been allocated into the correct buckets, the second step is to aggregate the total spending per customer. This process is shown in Figure 6-34. The execution of this SQL results in a single record per customer.

```
CREATE VIEW  ITSO.CUSTOMER_TOTALS AS

  SELECT

    CUSTOMER_ID AS CUSTOMER_ID,
    SUM(TOTAL_VISITS)                AS   TOTAL_VISITS,
    SUM(TOTAL_SPENT)                 AS   TOTAL_SPENT,
    SUM(TOTAL_ITEMS)                 AS   TOTAL_ITEMS,
    SUM(TOT_SPENT_CASH )             AS   TOT_SPENT_CASH,
    SUM(TOT_SPENT_CREDIT_CARD )      AS   TOT_SPENT_CREDIT_CARD,
    SUM(TOT_SPENT_DEBIT_CARD )       AS   TOT_SPENT_DEBIT_CARD,
    SUM(TOT_SPENT_JAN)               AS   TOT_SPENT_JAN,
    SUM(TOT_SPENT_FEB)               AS   TOT_SPENT_FEB,
    SUM(TOT_SPENT_MAR)               AS   TOT_SPENT_MAR,
    SUM(TOT_SPENT_APR)               AS   TOT_SPENT_APR,
    SUM(TOT_SPENT_ELECTRONICS)       AS   TOT_SPENT_ELECTRONICS,
    SUM(TOT_SPENT_COMPUTERS)         AS   TOT_SPENT_COMPUTERS

  FROM  ITSO.ALLOCATE_SPENDING_VIEW

  GROUP BY CUSTOMER_ID;
```

Figure 6-34 Aggregate customer spending: sample SQL script

Since the total spending is already summarized at the customer level, the third step is to calculate the percentages. This process is depicted in Figure 6-35. The target customer behavior table has the percentages variables defined as *decimal* data types. To obtain the target values in decimal, the calculation formulas transform all variables to decimal format so that results are produced with decimals. After the variables have been calculated, the final step is to insert the variables into the target table.

```
CREATE VIEW ITSO.CUSTOMER_BEHAVIOR_VIEW AS

  SELECT

      CUSTOMER_ID AS CUSTOMER_ID,
      TOTAL_VISITS AS TOTAL_VISITS,
      TOTAL_SPENT AS TOTAL_SPENT,
      TOTAL_ITEMS AS TOTAL_ITEMS,
      DEC(((DEC(TOT_SPENT_CASH,15,2)                / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_CASH,
      DEC(((DEC(TOT_SPENT_CREDIT_CARD,15,2)         / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_CREDIT_CARD,
      DEC(((DEC(TOT_SPENT_DEBIT_CARD,15,2)          / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_DEBIT_CARD,
      DEC(((DEC(TOT_SPENT_JAN,15,2)                 / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_JAN,
      DEC(((DEC(TOT_SPENT_FEB,15,2)  / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_FEB,
      DEC(((DEC(TOT_SPENT_MAR,15,2) / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_MAR,
      DEC(((DEC(TOT_SPENT_APR,15,2)  / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_APR,
      DEC((( DEC(TOT_SPENT_ELECTRONICS,15,2)        / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_ELECTRONICS,
      DEC(((DEC(TOT_SPENT_COMPUTERS,15,2)           / DEC(TOTAL_SPENT,15,2) ) * 100),5,2) AS PCT_SPENT_COMPUTERS

 FROM   ITSO.CUSTOMER_TOTALS ;


INSERT INTO ITSO.CUSTOMER_PURCHASE_BEHAVIOR
  SELECT * FROM ITSO.CUSTOMER_TOTALS;
```

Figure 6-35 Calculation of customer percentage spending: sample SQL script

All the calculations and transformations provided in the previous examples are performed using views instead of materialized tables. Using views is a good technique because it makes maintenance simpler. During the initial phases of implementation of a new data mining process, many experiments and several iterations of transformations may be required. The use of views to perform the transformations offers a very flexible approach to test new data mining models using different combinations of variables.

6.4.4 Performance considerations

Data preparation can be a very time-intensive activity. Depending on the complexity of transformations and volumes of data, this activity can account for well over half of the overall time in the data mining project. Because data mining projects by nature consist of many experiments using different settings for mining procedures and different combinations of variables, several iterations in the data preparation process could be required to generate new variables for data mining.

If dealing with very large tables, data preparation can be especially time-consuming. A good technique to minimize the time spend on this activity is to use a subset of the data. The process of using a subset of the data can be implemented by obtaining a random sample of the data. This can be done by using the *rand()* function (rand function) in the select statement to retrieve the source data. It can be easily implemented in a Design Studio data flow as well through SQL scripts. An example of data flowing with a where clause condition operation is shown in Figure 6-36. The condition property of this operator is defined as RAN(3) < 0.10, which generates a SQL statement that selects only 10 percent of the source data, as shown in Table 6-1 on page 138.

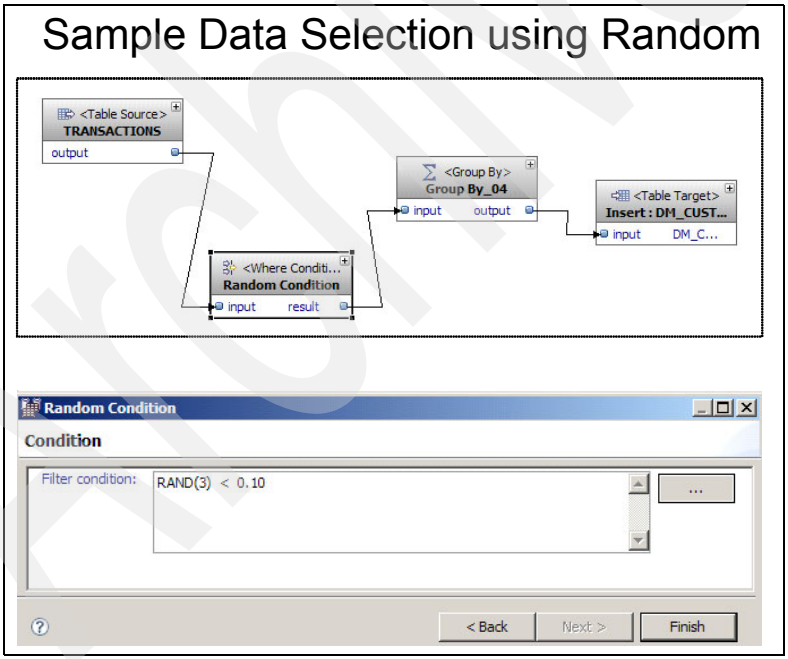


Figure 6-36 Aggregate customer spending: sample SQL script

Table 6-1 Sample random function

```
SELECT  ...
   SALES_AMOUNT AS SALES_AMOUNT,
   ...
   FROM
    BISW.TRANSACTIONS

   WHERE (RAND(3)  <  0.10;
```

The use of views to perform transformations provides flexibility during the development phase of the data mining models. After the models have been tested and validated, however, it might be worthwhile to analyze carefully the performance of data retrieval against views versus materializing the transformations into a table. In particular, on large databases this activity can consume large amounts of systems resources such as CPU and memory. By materializing the transformations into a regular table, the system's utilization of resources during model refreshes can be reduced.

7

Data Mining in DB2 Warehouse

Data mining is performed in DB2 Warehouse by creating and executing mining flows through the Design Studio interface. In this chapter, we provide an example of how to create mining flows and interpret results for each of the data mining techniques in DB2 Warehouse. For further discussion of mining flows and model interpretation, see *Leveraging DB2 Data Warehouse Edition for Business Intelligence*, SG24-727474. Appendix A, “DB2 Warehouse data mining algorithms: a deep dive” on page 331 contains details of the algorithms represented in the mining flows discussed in this chapter.

7.1 Mining flows

A mining flow is used to design and execute a mining process. Every mining flow includes at least one input table and a mining operator specific to the mining technique being used. SQL functions such as table joins, column selections, sampling, filters, and other data preparation functions are often included in a mining flow. Most mining flows also have one or more output targets such as a visualizer operator or an output table containing scored results. Scoring may be done in the same flow as the modeling or in a separate flow using a mining model as an input.

The following sections of this chapter use examples to illustrate the basics of designing and executing mining flows for the five data mining methods in DB2 Warehouse. In each example, the mining flow shows typical steps to create a model, validate it (in the case of predictive models), and apply it to score new records. Although scoring is not required to build or validate a model, the scoring step may be used to add the mining results to the input data for further analysis such as associations or sequences analysis for a selected customer segment.

Mining flows are developed in the Business Intelligence (BI) perspective in the Design Studio, illustrated in Figure 7-1. See *Leveraging DB2 Data Warehouse Edition for Business Intelligence*, SG24-72744 for additional information about mining flow creation, validation, and execution.

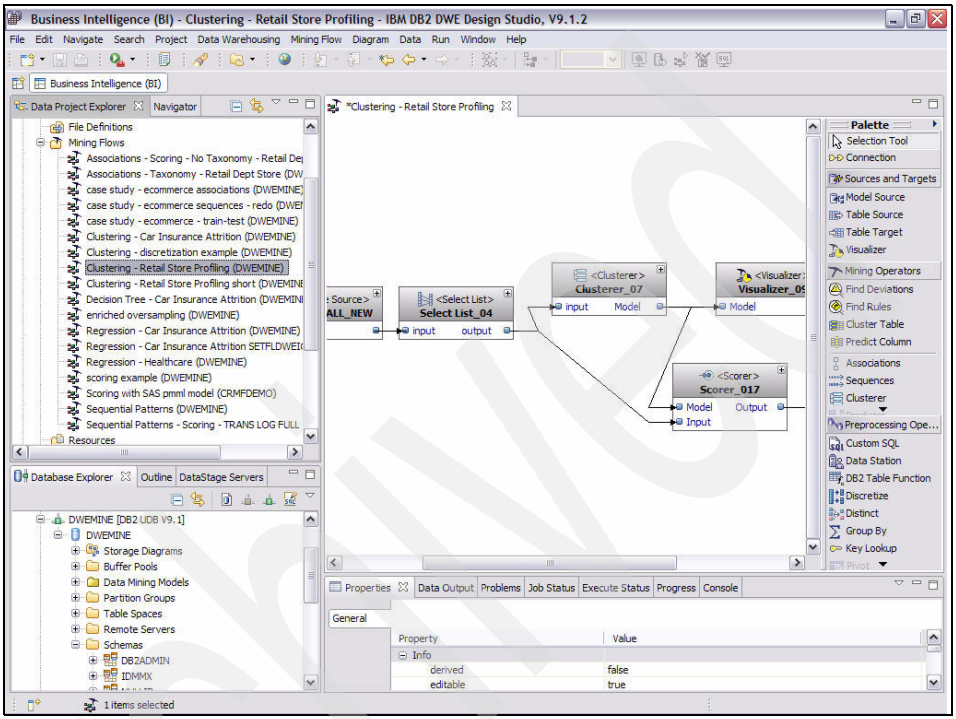


Figure 7-1 Business Intelligence perspective in the Design Studio

Data mining models created in mining flows are stored in four tables under the IDMMX schema. These tables are created when a database is enabled for data mining. Mining models are expressed in the Predictive Modeling Markup Language (PMML) format, which is an XML-based language for defining statistical and data mining models so that they can be shared and used by PMML-compliant applications.

7.2 Clustering

The following example illustrates the clustering method. The business problem is to design the layouts (proportion of floor space allocated to each department) of several new stores that a nationwide retail department store chain is planning to build in the southern U.S. Specifically, the objective is to determine which departments should be emphasized or de-emphasized as a percentage of total store floor space to best serve climate-dependent purchasing preferences of the customers in the target region. This problem is common in the retail industry and can be approached as an application of clustering to perform an analysis called *store profiling*.

For store profiling analysis, variables are needed (fields or columns in a table) representing each store's revenue by department as a percentage of that store's total revenue, thereby creating a sales performance measure for each department. For a given store, if the percentage sales for a particular department is greater (less) compared to all stores, then that store is considered to overperform (underperform) in that department. (For a retail customer segmentation, the analog would be the percentage spending by department, along with demographic attributes if available. For a banking customer segmentation, the analog would be the percentage of total deposits by account type.)

7.2.1 Building a clustering mining flow

The mining flow for this clustering process is shown in Figure 7-2 on page 143. The flow begins with an input Table Source operator (1). The source table specified in this operator contains one record per store with columns representing sales performance by department and store demographics (for example, climate and how long the store has been open). Since some of the columns in the source table are not meaningful for characterizing store behavior, a Select List operator (2) is used to select a subset of the columns. The selected columns flow into a Clusterer operator (3) where parameters and other settings for building the mining modeling are specified. The model is piped to a Visualizer operator (4), which displays the model results visually. To generate an output table containing the scoring results (cluster number and other information for each input record), the input table and the model are passed to a Scorer operator (5) which applies the model to the input records to generate the scoring information and then write the results to an output table specified in a Table Target operator (6).

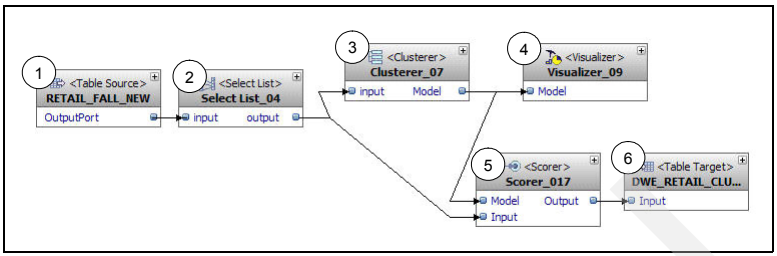


Figure 7-2 Mining flow for clustering

The Table Source operator (1) contains a useful parameter called the sampling rate, as shown in Figure 7-3 on page 144. This parameter is usually left at 100 percent, meaning that all records in the source table will be used. Setting it to a lower value will result in a random selection of the source records being passed to the next step in the mining flow. This feature is useful, for example, when the source table contains a large number of records. Since modeling is often a repetitive process of trying various sets of variables (columns in the input table) in conjunction with mining parameters to find the best model, using a small portion of the records for preliminary model development will reduce execution time. For example, if the source table contains 10 million records, then setting the sampling rate to 5 percent will extract a random sample of 500,000 to build the preliminary model. The sampling rate can then be reset to 100 percent for the final execution of the mining flow.

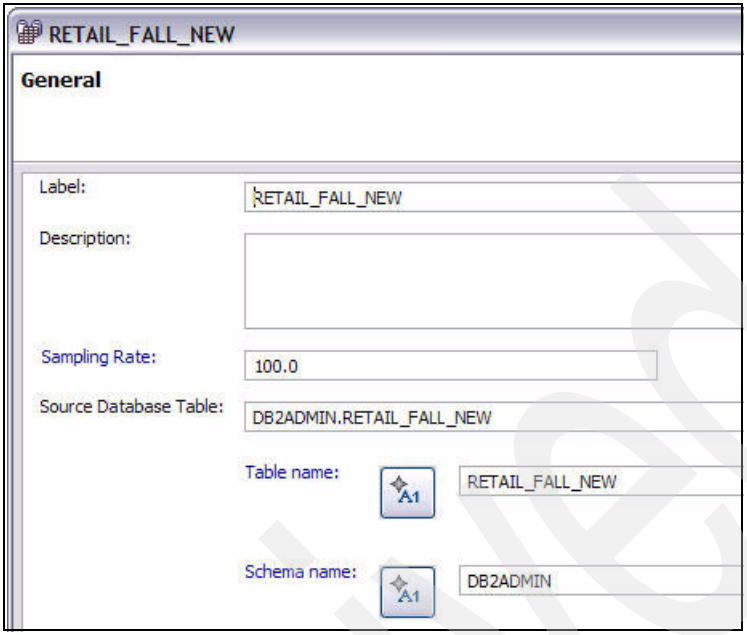


Figure 7-3 Table Source operator: setting the sampling rate

Mining settings that govern the creation of the clustering model are specified in the Clusterer operator (3), as shown in Figure 7-4 on page 145. Here, the maximum number of clusters is set to 10. As a rule of thumb, setting this parameter to a value of about 8 to 20 usually produces a reasonable number of clusters. Too small a number may result in a few clusters that are not very distinct from one another, while too large a number may yield some clusters that are too small to be meaningful for making a business decision. In this example, an optional parameter is specified to create and apply a name mapping that generates descriptions for the input values of climate (for example, 1=Warm) to facilitate interpreting the clustering results. In most cases, however, the Optional Parameters field is left blank. The demographic clustering algorithm is selected, with a similarity threshold of 0.5. Specific to this algorithm, the similarity threshold defines how similar records must be to be allowed in the same cluster. The default value of 0.5 indicates that records in the same cluster must be similar across at least 50% of the attributes (variables in the model, represented by columns in the input table). If the clustering model consists of one very large cluster and the rest small, then setting the similarity threshold to a higher value (for example, 0.6 to 0.9) tends to yield a less uneven distribution of record counts across the clusters.

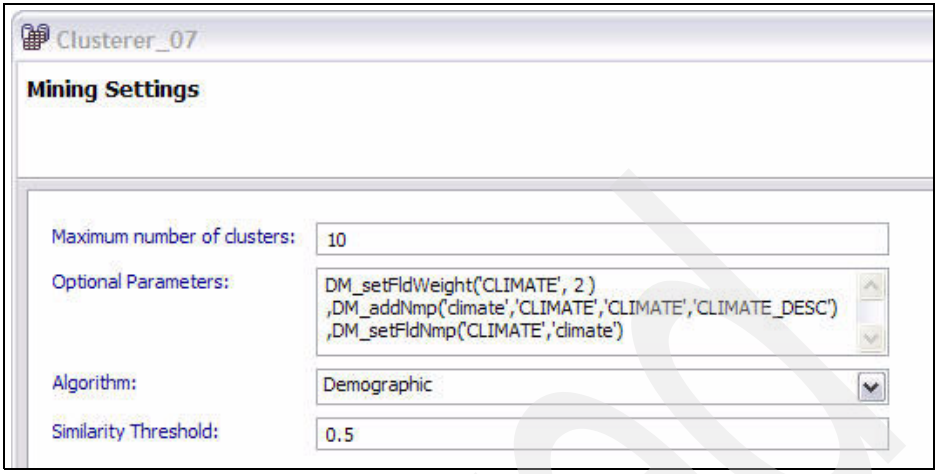


Figure 7-4 Clusterer operator: specifying mining settings

As an alternative to the demographic clustering algorithm, a neural clustering algorithm called the Kohonen self-organizing feature map can be selected from the Algorithm drop-down list, as shown in Figure 7-5 on page 146. Instead of the similarity threshold, the number of passes that the algorithm will make as it converges on a model is specified. A value of 10 generally is sufficient, with larger values not improving the model very much. If the number of records is large, however, then about five passes could be used during the repetitive process of creating a satisfactory model (to reduce execution time) and then increased to 10 or 20 for the final run.

In many cases, depending on the data, Kohonen neural clustering may give more homogeneous clusters and thus better model quality than demographic clustering. As a rule of thumb, Kohonen clustering tends to give better results when most or all of the variables are numeric.

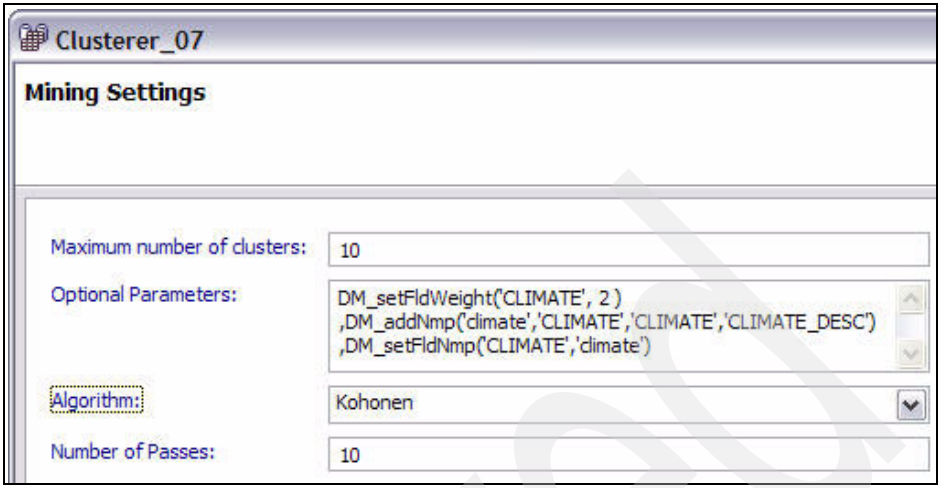


Figure 7-5 Clusterer operator: selecting a different algorithm

Treatment of the variables (fields) in a model can be controlled by specifying column properties, as shown in Figure 7-6 on page 147. Under the Field Usage Type column, the default is System Determined, which allows the mining kernel to determine how to use each field in a model. For example, the kernel automatically calculates correlations between pairs of fields and may exclude one or more fields that are highly correlated with others to avoid "double-counting" of effects (using Pearson's correlation for numeric fields and chi-square contingency for categorical fields). It also excludes fields that have a distinct value for each record, considering them to be keys rather than explanatory variables. It also excludes fields that are single-valued (or nearly so) or that have missing values for most records. Second, a variable can be specified as Active to ensure that it is used in building a model. Third, a variable can be specified as Supplementary so that it is not used in building the model but is displayed in the clusters. This option is useful when, for example, the historical data used to build the model contain an attribute that would not be known for new records, for example, whether or not a cardiac patient has another heart attack.

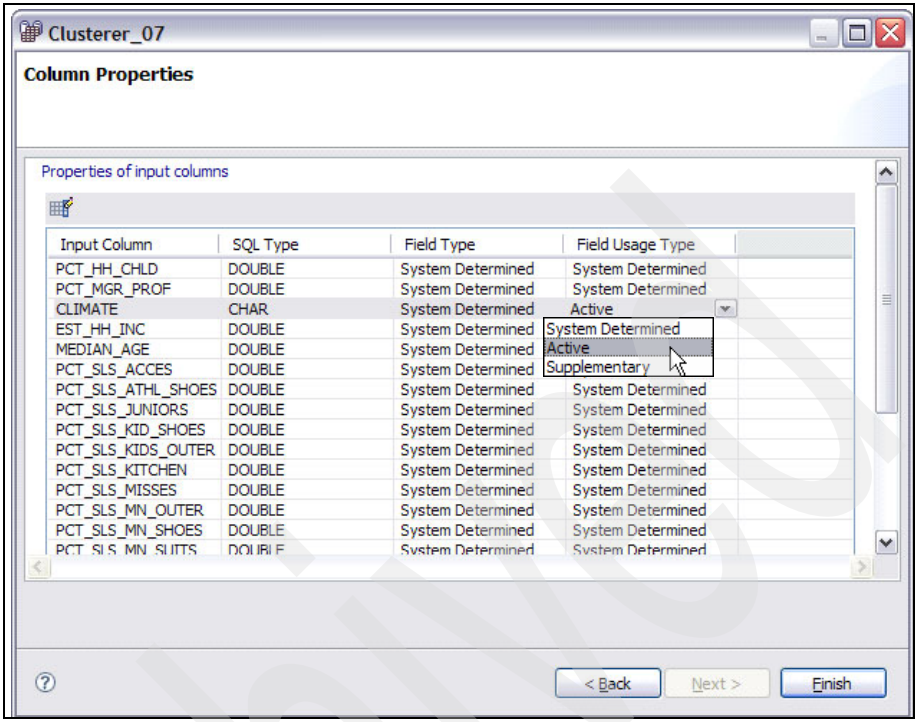


Figure 7-6 Clusterer operator: specifying column properties

The Scorer operator (5) applies the model to a source table and writes out the scored results to a target table (6). The source table may be a new set of records or, as in the example shown in Figure 7-2 on page 143, the source may be the same table as used to build the model. In this example, the Scorer serves to make the modeling results accessible in an output table; no new records are being scored.

The Scorer allows the user to specify which input columns are written to the target table along with the scoring results. The target table should include at least the record identifier and the scoring results, but often the input columns are also included to facilitate other analysis such as OLAP. For clustering, the scoring results for each record include the cluster ID, the quality of the fit of the record to its assigned cluster, and the confidence that the record has been placed into the correct cluster. The Add, Edit, and Delete buttons on the Result Items wizard view of the Scorer can be used to display the desired result columns with the preferred names (aliases), as shown in Figure 7-7.

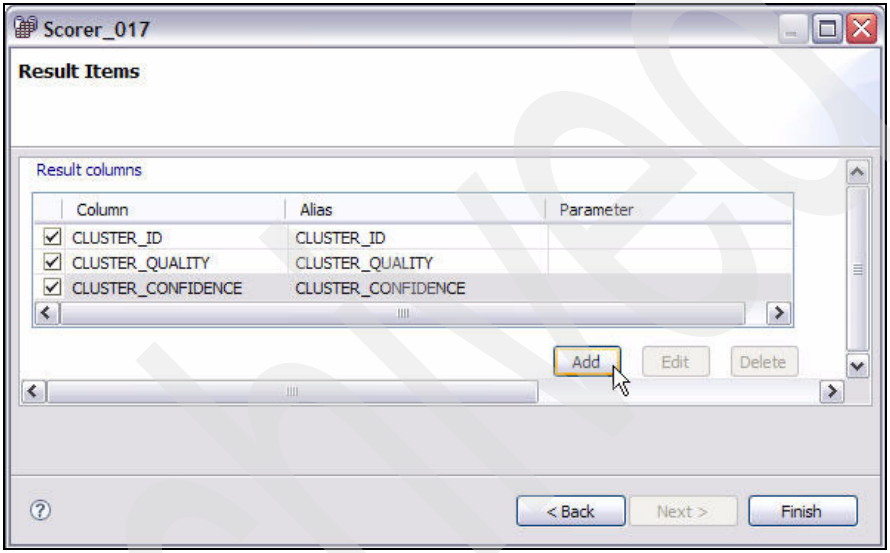


Figure 7-7 Scorer operator: specifying scoring results

The target table that receives the scoring results from the Scorer is created by right-clicking the output port of the Scorer and selecting **Create Suitable Table** to open a wizard, as shown in Figure 7-8.

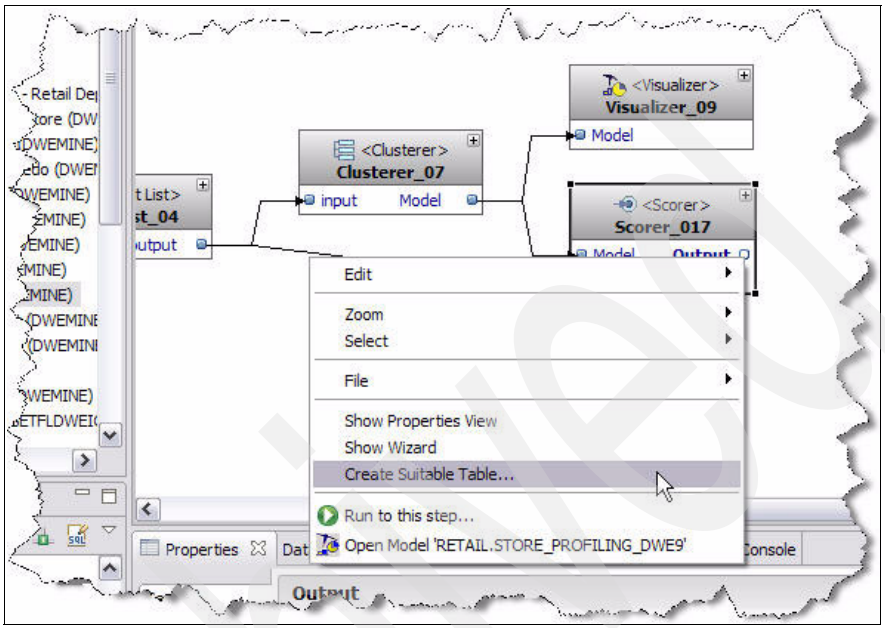


Figure 7-8 Scorer operator: creating a suitable output table

The analyst specifies a name for the new table, and the Table Target operator (6) is created and automatically placed into the mining flow, as shown in Figure 7-9.

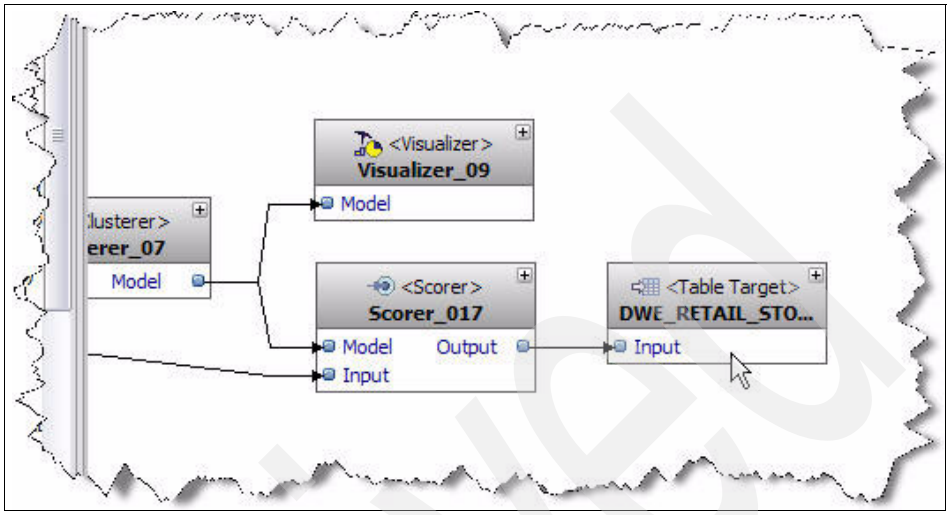


Figure 7-9 Table Target operator - creating a target table

Double-clicking the **Table Target operator** in Figure 7-9 will open a properties window for the output table, as shown in Figure 7-10 on page 151. Typically, the box to **Delete previous content** should be checked to ensure that new output records replace the old records, rather than being appended, when the mining flow is re-executed. This selection truncates the output table, meaning that the old records are deleted without deleting the table itself. For more information about table truncation, see 7.8.2, “Clearing a full DB2 transaction log” on page 211.

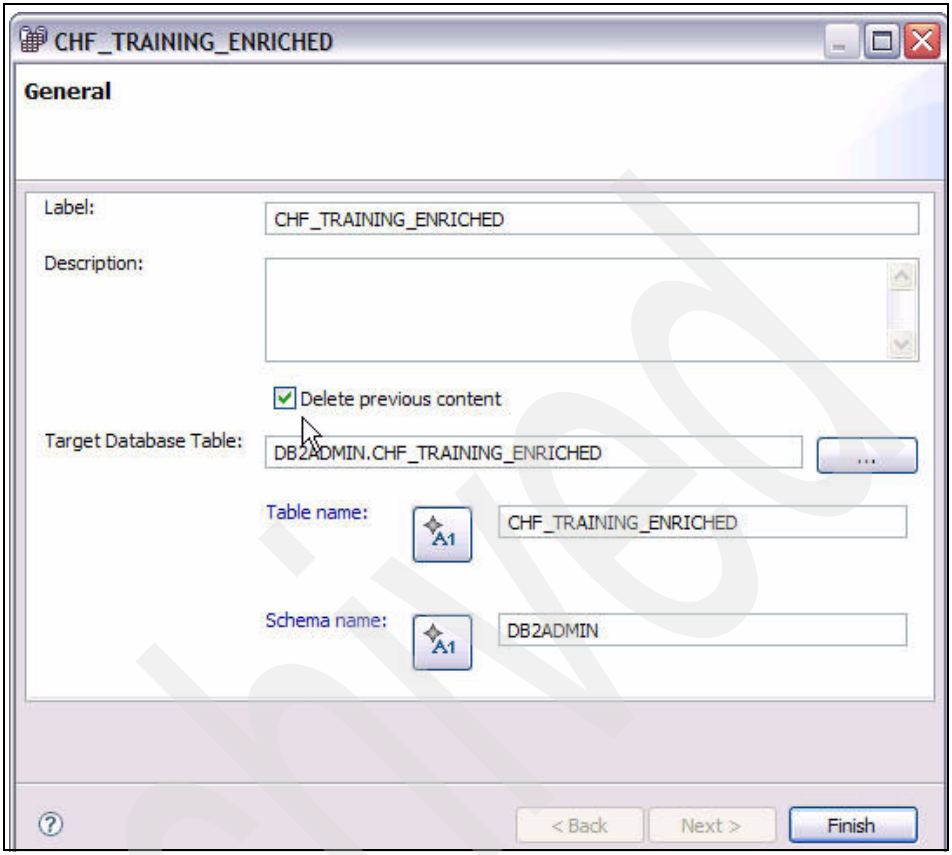


Figure 7-10 Table target: Delete previous content

This completes the mining flow as shown in Figure 7-2 on page 143. The flow is executed by selecting **Mining Flow** → **Execute** from the Design Studio menu bar. Upon completion, a visualization of the model generated by the mining flow is automatically displayed.

7.2.2 Interpreting a clustering model

The main view of the visualization is shown in Figure 7-11. The clusters are displayed as rows sorted by size (percentage of stores in each cluster). Within each cluster, the variables are sorted in descending order of importance (as measured by a chi-square statistic) in distinguishing the members of that cluster from all other stores. Because the objective of clustering is to segment individuals into homogeneous groups that are different from other groups, we interpret clusters by comparing each variable's distribution for a given cluster to its distribution for all stores.

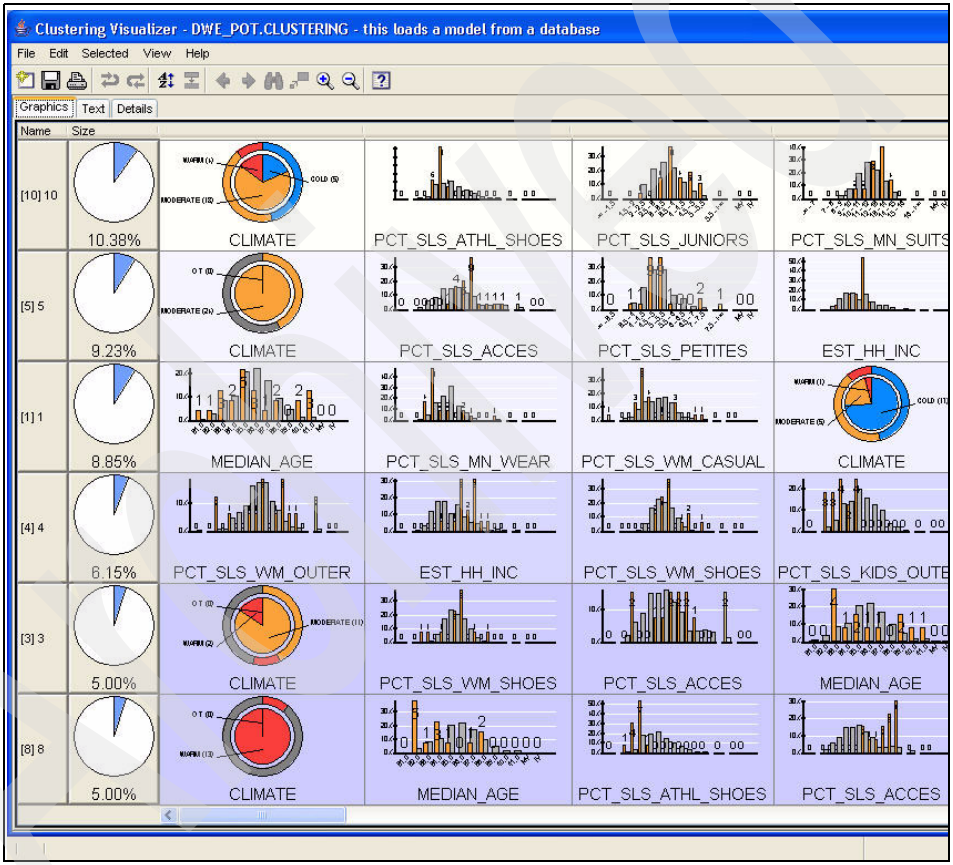


Figure 7-11 Clustering visualization example

For a categorical variable, the distributions are represented by concentric pie charts, as illustrated for a variable called CLIMATE in Figure 7-12. The inner circle represents a particular cluster, while the outer ring represents all stores as a whole. In this example, we see that 67% of the stores in this cluster are located in moderate-climate cities, compared to 43% of all stores. Fifteen percent of the stores in this cluster are located in warm-climate cities, compared with 11% of all stores, and only 18% are in cold-weather cities compared to 46% of all stores. Thus, we interpret this cluster as being comprised primarily of stores in moderate and warm climates, in contrast to all stores in the nationwide chain.

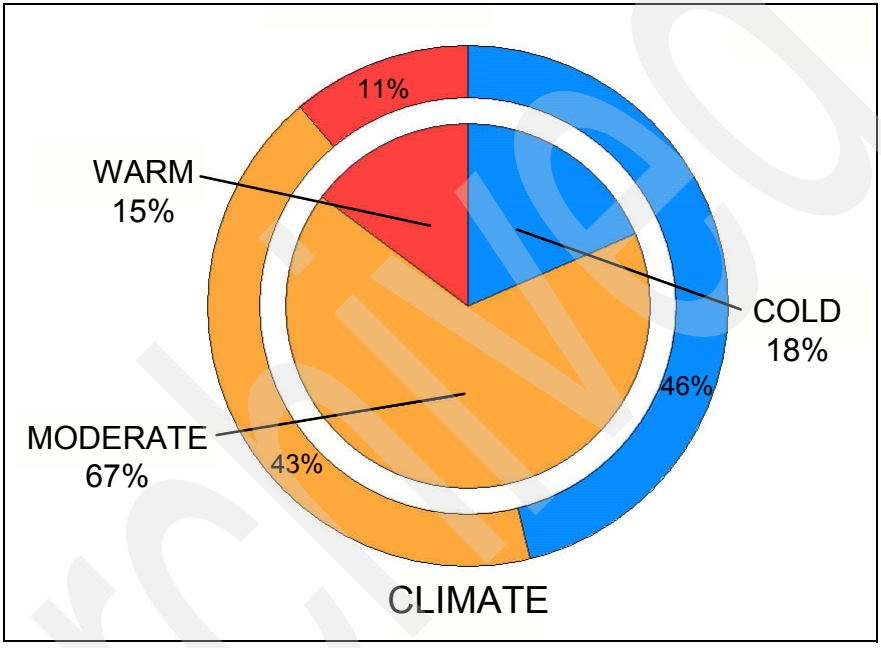


Figure 7-12 Representation of distributions of a categorical variable

For a numeric variable, the distributions are represented by percentage histograms, as illustrated for a variable called PCT SALES ATHL SHOES in Figure 7-13. The foreground histogram (consisting of three bars labeled 22%, 63%, and 15%) represents a particular cluster, while the background histogram represents all stores as a whole. In this example, we see that the stores in this cluster derive, on average, about 7 to 8% (that is, between 6% and 9% as shown by the three bars) of their revenue from this department, compared to an average of around 8 to 9% for all stores (the median of the background histogram). Thus, we interpret this cluster as being comprised of stores that, on average, derive less of their revenue from this department (that is, underperform), in contrast to all stores.

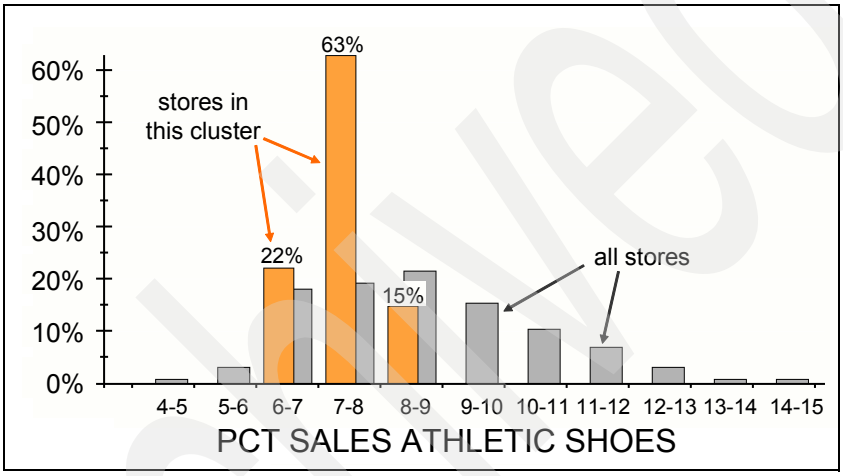


Figure 7-13 Representation of distributions of a numeric variable

When examining a clustering model, differences among the clusters can readily be discerned by visually comparing the distributions of the most important (highest-ranked) variables across clusters. The relative sizes of the inner (this cluster) and outer (all records) values for categorical variables and the relative positions of the foreground (this cluster) and background (all records) histograms for numeric variables across the top several variables provide a readily interpretable overview of the distinct characteristic profiles represented by the clusters. In addition, the variables can be ordered in various ways, for example, alphabetically or by average values of a particular variable, or displayed as pie charts, histograms, or tables to assist with interpreting the model. Many other options, such as colors and chart displays, are available as well.

In this example, we are interested in the expected purchasing patterns for the planned stores. We search the clusters to find the one that appears to best represent the new stores and hence can serve as a pattern for allocating floor space among departments. Retailers know that climate is a strong predictor of customers' purchasing behavior, so we look for a cluster having a predominance of warm-climate stores. In this case, Cluster 8 (the last cluster in Figure 7-11 on page 152) meets this criterion as well as other relevant criteria such as relatively new stores, as shown in Figure 7-14. (Note that labels have been added to assist with interpreting this cluster.) In terms of the business question, we see that stores in Cluster 8 tend to under perform in certain departments (labeled *Low performance*) and to over perform in other departments (labeled *High performance*) relative to other stores, as explained for Figure 7-13 on page 154. These characteristics indicate which departments to emphasize or de-emphasize in allocating floor space.

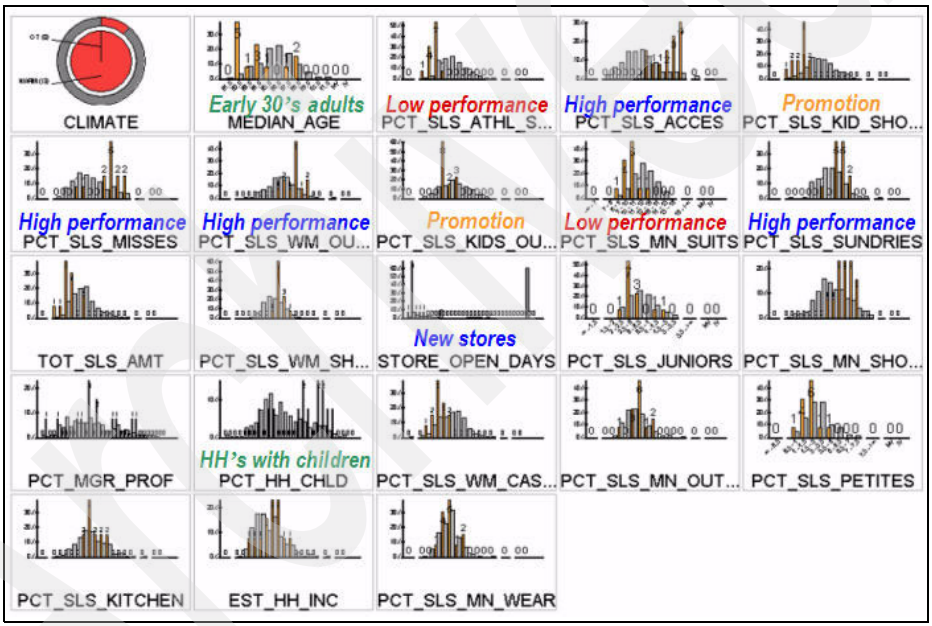


Figure 7-14 Clustering visualization example: drill down to one cluster

As is often the case with data mining, we may find other opportunities in addition to answering the primary business question. In this case, we see from the demographic attributes in Figure 7-14 that the customer base for the stores in Cluster 8 consists mostly of adults in their early 30s and households with children and relatively high household income. This finding suggests promotional opportunities in departments such as children's shoes and children's outer wear. Other clusters may also reveal valuable marketing and merchandising opportunities not directly related to the primary business question.

7.2.3 Scoring with a clustering model

When a clustering model is applied to one or more records, the Scorer assigns a cluster ID, a quality value, and a confidence value to each individual record being scored. The quality and confidence value are different measures that indicate how well the record fits into the assigned cluster. The quality and confidence values both range from 0 to 1. A quality value near 0 means that a particular record fits very poorly into its assigned cluster, while a value of 1 means that the record fits perfectly into the cluster. A confidence value near 0.5 means that the record fits into another cluster about as well as the assigned cluster, while a value near 1 indicates that the record fits only its assigned cluster. The scored cluster ID can then be used, for example, to send a personalized mailing to those customers with a specific cluster ID value based on the characteristics of this cluster in light of the business objective.

The meanings of quality and confidence are illustrated in Figure 7-15. In Case 1, Record X is assigned to Cluster 1 with high confidence (since there is no competing cluster) and high quality (Record X has a good fit with Cluster 1). In Case 2, Record Y is assigned to Cluster 1 with high confidence (no competing cluster) but low quality (poor fit with Cluster 1). In Case 3, Record Z is assigned to Cluster 1 with low confidence (competition from Cluster 2) but high quality (good fit with Cluster 1).

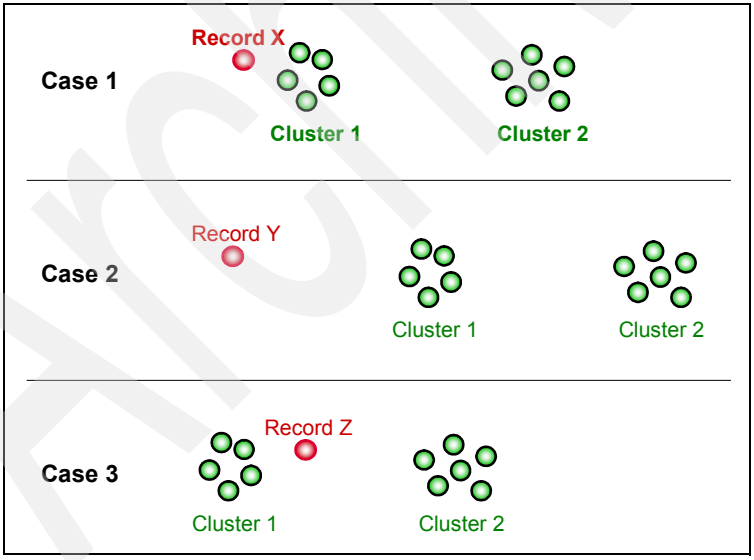
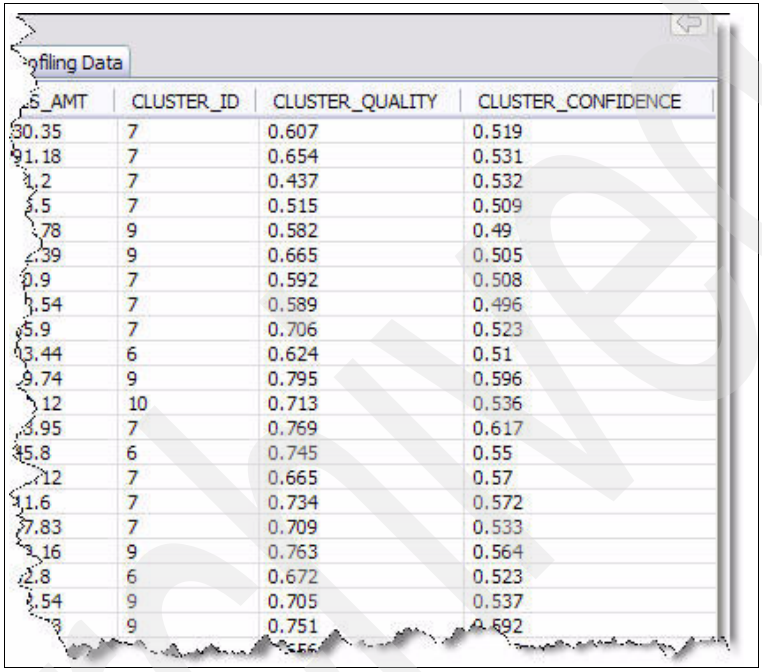


Figure 7-15 Clustering quality and confidence

A sample of the scored results from the output table generated by the scorer (Figure 7-2 on page 143) is shown in Figure 7-16. For example, the first record has been assigned to Cluster 7 with a quality of 0.607 and confidence of 0.519. These values suggest that the record fits fairly well to its assigned cluster with reasonable (although not very high) confidence. Depending on how the output table is specified, it may contain some or all of the input columns used as variables in the clustering model as well as the scoring results.



filing Data

S_AMT	CLUSTER_ID	CLUSTER_QUALITY	CLUSTER_CONFIDENCE
30.35	7	0.607	0.519
91.18	7	0.654	0.531
1.2	7	0.437	0.532
3.5	7	0.515	0.509
78	9	0.582	0.49
39	9	0.665	0.505
0.9	7	0.592	0.508
3.54	7	0.589	0.496
5.9	7	0.706	0.523
3.44	6	0.624	0.51
9.74	9	0.795	0.596
12	10	0.713	0.536
3.95	7	0.769	0.617
15.8	6	0.745	0.55
12	7	0.665	0.57
11.6	7	0.734	0.572
7.83	7	0.709	0.533
16	9	0.763	0.564
2.8	6	0.672	0.523
1.54	9	0.705	0.537
3	9	0.751	0.692

Figure 7-16 Clustering: scoring results

7.3 Associations

The following example illustrates the associations method. A transaction, or item set, is a collection of one or more individual items. Associations analysis is a method to find relationships (affinities) among items within transactions.

In this example, the business problem is to boost sales of women's casual wear merchandise for a retail department store. The objective is to find affinities between women's casual wear items and other items around the store that can be leveraged in a promotion or used for making cross-selling offers. This problem can be approached as an application of associations to perform an analysis that is sometimes called a *market basket analysis* (MBA).

7.3.1 Building an associations mining flow

For associations analysis, we need a transactions table containing columns representing the transaction identifier for each transaction and the items contained in that transaction. Optionally, we need one or more tables containing item descriptions (to facilitate interpretation of the mining results), a hierarchical structure (taxonomy) relating items to higher levels of organization (for example, group, department, or division), and descriptions of the elements of the taxonomy. Depending on the business problem, each item may represent a product, event, characteristic, symptom, or outcome.

The mining flow for associations modeling is shown in Figure 7-17 on page 159. The mining flow is constructed as follows:

- As usual, the flow begins with an input Table Source operator (1).
- Because the source table contains some items not needed for associations analysis, a Select List operator (2) is used to select the needed subset of the columns.
- The selected columns flow into the data Input port of an Associations operator (8) where parameters and other settings for building the mining modeling are specified.
- Another Table Source operator (3) providing a mapping of product IDs to product descriptions flows into a Names port of the Associations operator.
- Because this model employs a taxonomy with two levels above the product level (Product < Group < Department), two additional Table Source operators flow into additional Names ports to provide name mappings of group ID and group description (4) and department ID and department description (5).
- Additional Table Source operators define the first level (product ID to group ID) and second level (group ID to department ID) of the taxonomy (6 and 7, respectively). Note that, depending on how the source tables are structured, a particular table may serve as the source for more than one operator, for example, MAIL_NAMES is used both for the product name mapping and for the product-group taxonomy.
- The mining model created according to the specifications and parameters in the Associations operator (8) flows to a Visualizer operator (9) that displays the model results graphically.
- The model also flows to an Associations Extractor operator (10) that extracts the association rules (expressions of the relationships among products, groups, and departments).

- The Associations Extractor operator (10) sends the extracted rules to a Table Target operator (11) that writes the rules to a DB2 table. The Table Target operator is created from the Associations Extractor operator using the Create Suitable Table function, as explained previously with regard to Figure 7-8 on page 149 and Figure 7-9 on page 150.
- Another Table Source operator (12) contains new orders that will be scored by the associations rule model.
- A Select List operator (13) selects the product ID and the transaction ID and pipes these two columns to the Scorer (14).
- The Scorer operator (14) reads in the model from (8) and applies it to the new transactions in (12). The scored results are written to a new table defined by a Table Target operator (15) using the Create Suitable Table function.

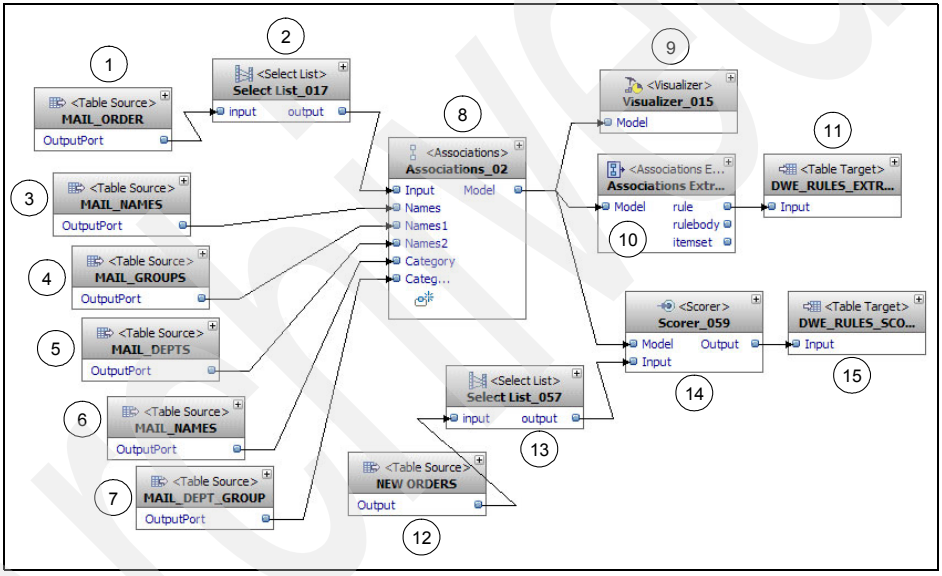


Figure 7-17 Mining flow for associations

Although the Table Source operator allows a sampling rate between 0 and 100 percent, sampling should not be done this way for associations mining. Because a transactions table contains multiple records per transaction, selecting a random sample of the records would not preserve the character of the transactions and therefore would not lead to valid associations rules. For an example of how to sample transactions properly, see 7.3.4, “Sampling transactions” on page 175.

The mining settings in the Associations operator (8) are shown in Figure 7-18 on page 161. Parameters are specified as follows:

- The group column refers to the input table column representing the transaction ID.
- The maximum rule length parameter determines the maximum number of items that an associations rule may contain. This parameter must be either 0 or greater than or equal to 2. For example, a maximum rule length of 4 means one item in the head (always) and up to three items in the body of a rule, while a value of 0 means that the number of items in a rule is unlimited.
- The minimum confidence parameter, expressed as a percentage from 0 to 100, sets the minimum percentage for how frequently the head item occurs among all the transactions containing the body items, that is, the reliability of a rule.
- The minimum support parameter, expressed as a percentage from 0 to 100, sets the minimum percentage of transactions that contain all of the items (head and body) listed in an association rule, that is, the prevalence of a rule. If a value of 0 is specified, then the value for minimum support is automatically determined to produce a result that contains at least some rules.
- The number of bins specifies the number of buckets (bins) automatically created for numerical data used as items in sequences or associations. The minimum value is 2 with no upper limit and a default value of 5.
- Filters and other constraints can be specified as optional parameters, although associations rules can also be filtered after the model has been created.

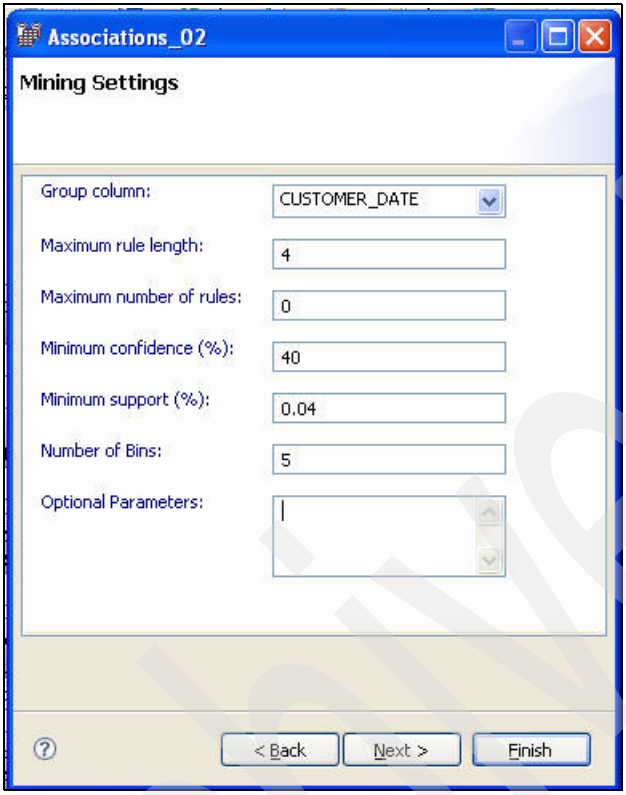


Figure 7-18 Associations mining flow: specifying mining settings

Name maps relate IDs to descriptions to make the rules easier to interpret. As illustrated in Figure 7-19, name maps can be created for Item ID as well as levels in a taxonomy.

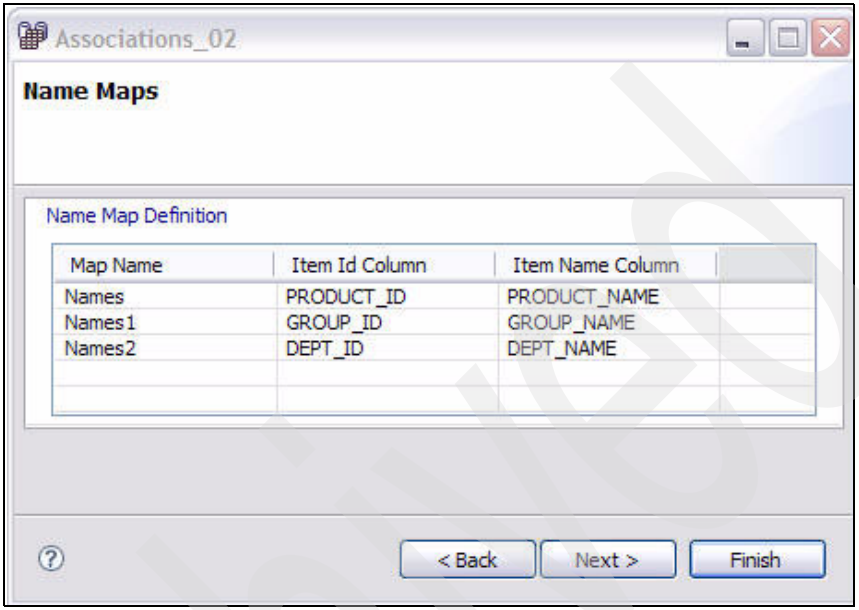


Figure 7-19 Associations mining flow: specifying name mappings

If used in an associations model, the taxonomy is defined by category maps, as shown in Figure 7-20. In this example, values of the Item ID (or Product ID) are mapped to corresponding values of the taxonomic level Group ID, while values of Group ID are mapped to corresponding values of the highest taxonomic level, Dept. ID. Here, the category maps are non-recursive, meaning that a map can hold relations only between two consecutive levels of a hierarchy, as compared to a recursive category map that can hold relations between more than two consecutive levels.

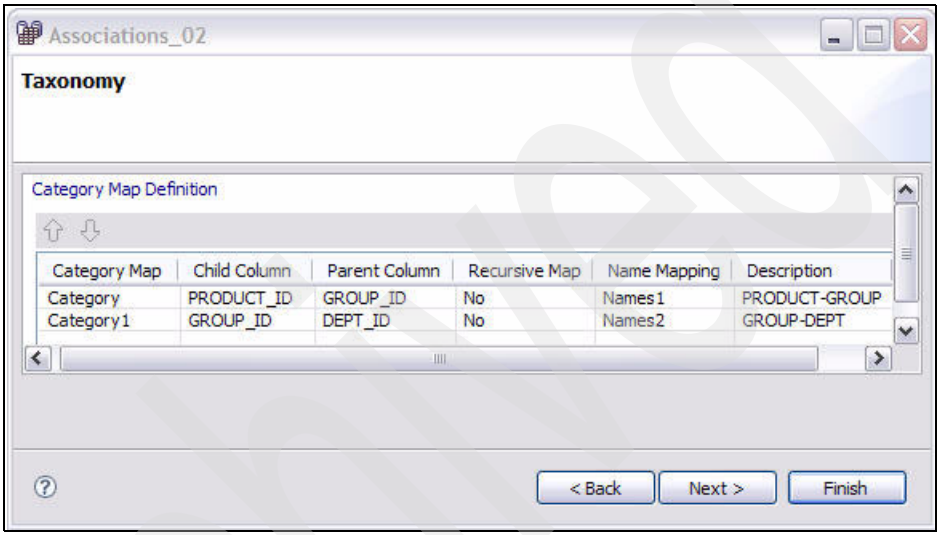


Figure 7-20 Associations mining flow: defining the taxonomy

An example of non-recursive and recursive category maps is shown in Figure 7-21, where the taxonomy can be represented either by the two non-recursive category maps 1 and 2 or, equivalently, by the single recursive category map 3.

Category map 1: Non-recursive

Child	Parent
Orange Juice	*Beverages*
Beer	*Beverages*
Tomato	*Vegetables*
Cabbage	*Vegetables*
Carrot	*Vegetables*

Category map 2: Non-recursive

Child	Parent
Beverages	*Food*
Vegetables	*Food*

Category map 3: Recursive

Child	Parent
Orange Juice	*Beverages*
Beer	*Beverages*
Tomato	*Vegetables*
Cabbage	*Vegetables*
Carrot	*Vegetables*
Beverages	*Food*
Vegetables	*Food*

Figure 7-21 Example of non-recursive and recursive category maps

The column properties are specified as shown in Figure 7-22. Columns other than the group column (transaction ID) are shown as input columns, so the product ID is shown here. If any unneeded columns appear, their field usage type can be set to “inactive” to ignore them. The input data (field) type can be changed to categorical if necessary. The name map for the item or product ID is specified for the name mapping. Whether or not to use the taxonomy is specified; this setting is useful when a taxonomy exists but the analyst wants to compare results with and without the taxonomy. If available, a weight column representing price or cost can be applied to the item column to weight certain items more heavily in the rules, enabling calculations such as potential revenue or relative costs of errors. The field usage type of a weight column should be set to “weight.”

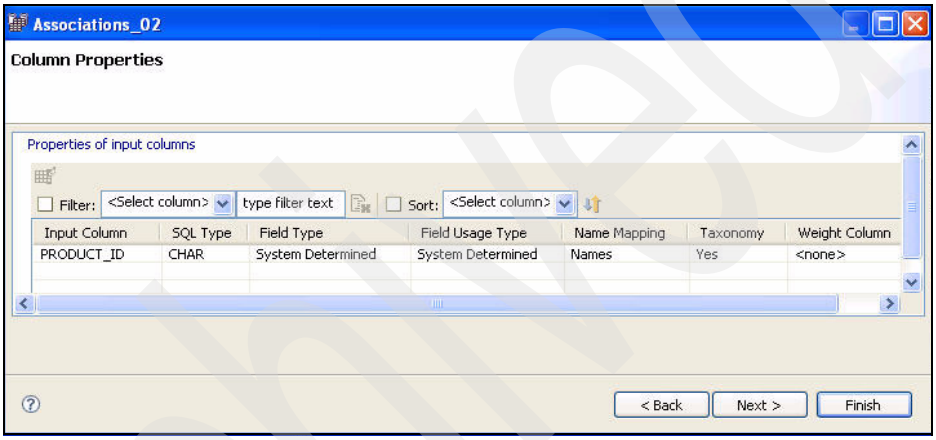


Figure 7-22 Associations mining flow: specifying column properties

Rule filters can be set up to limit the associations analysis to selected items or categories, as shown in Figure 7-23.

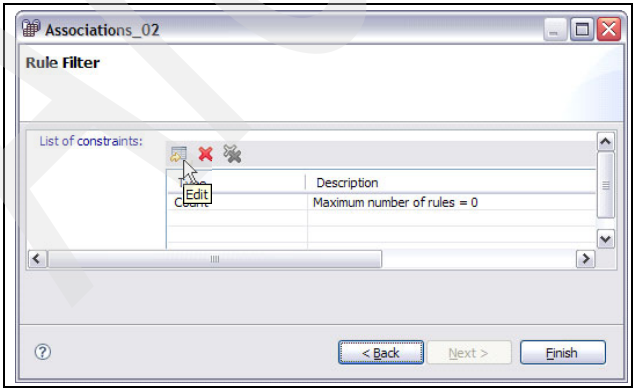


Figure 7-23 Associations mining flow: specifying rule filters

Clicking on the **Edit** button of the rule filter wizard accesses the settings for various filters, such as the item filters expression builder illustrated in Figure 7-24. In this example, double-clicking PRODUCT_ID populates the categories and values, which are selected and incorporated into a filter using the expression builder.

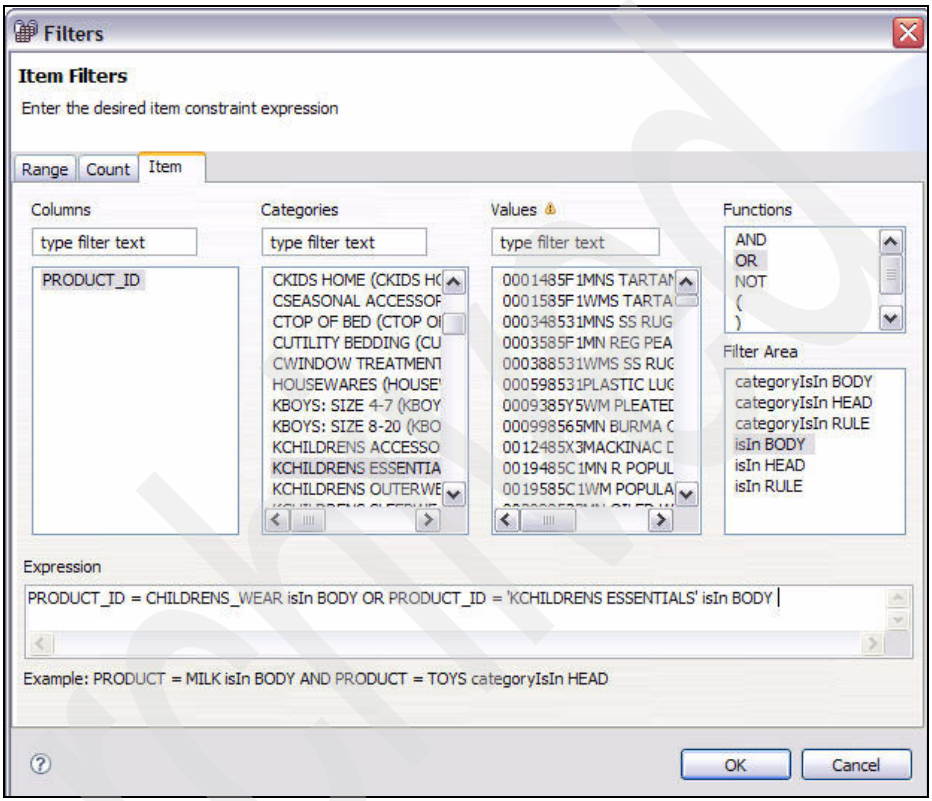


Figure 7-24 Associations mining flow: creating item filters

In the Scorer operator, the input columns are mapped to the mining model fields by dragging and dropping the input column names from the available input columns list on the left to the mapping list on the right, as shown in Figure 7-25.

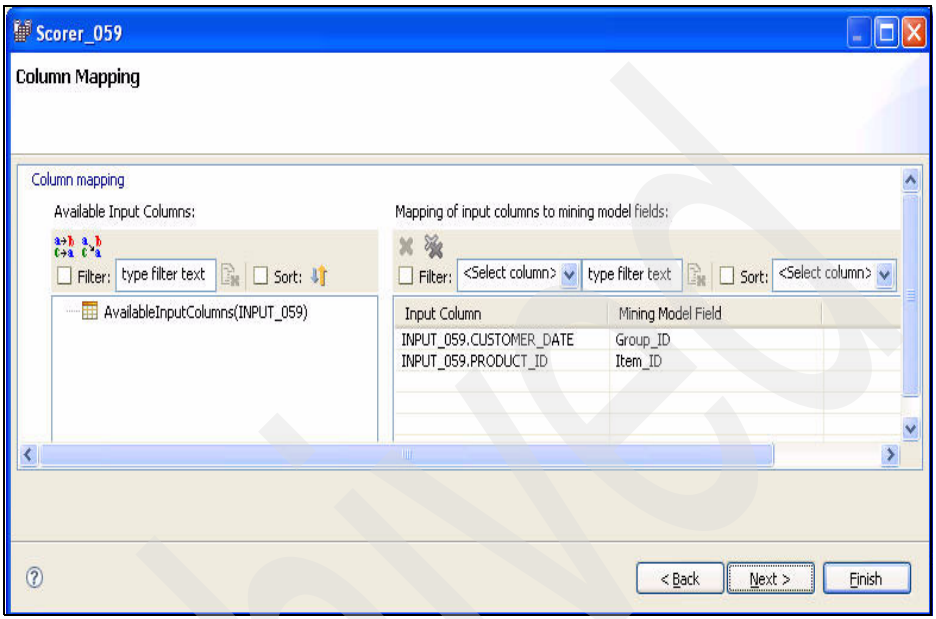


Figure 7-25 Associations mining flow: column mapping in the Scorer

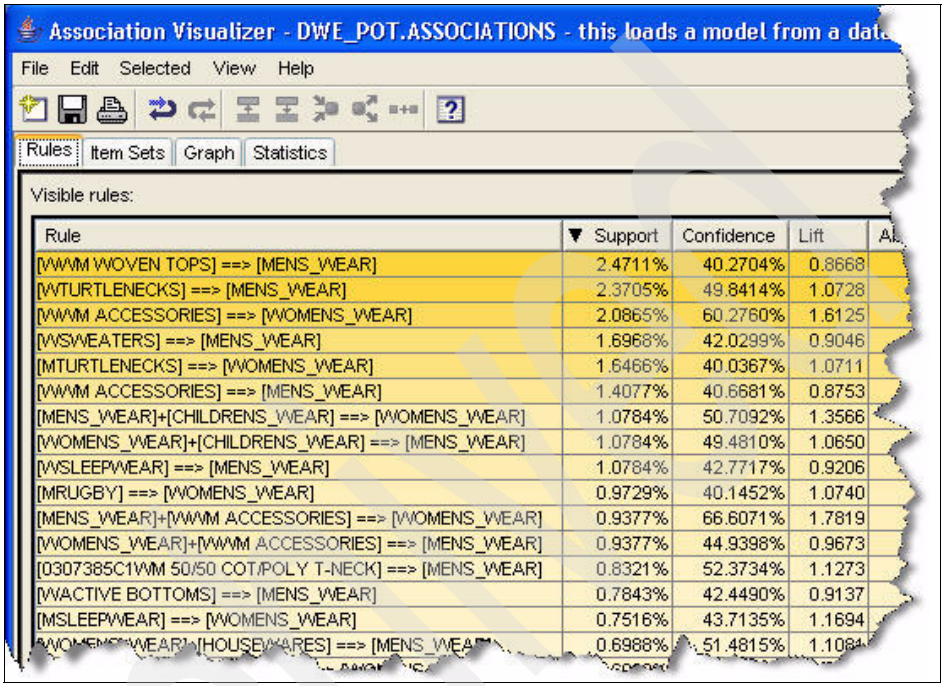
This completes the mining flow shown in Figure 7-17 on page 159. The flow is executed by selecting **Mining Flow** → **Execute** from the Design Studio menu bar. Upon completion, a visualization of the model generated by the mining flow is automatically displayed.

7.3.2 Interpreting an associations rule model

The associations visualizer presents a set of rules describing affinities among items or higher levels of taxonomy that occur together in a transaction. The associations visualization includes tables of affinity rules, the item sets on which the rules are based, a graph of the rules, and a table of statistics about the transactions data, mining parameters, and rules. The graph and tables are interactive to aid the analyst in focusing on rules of interest for a particular business objective. For example, if an upcoming promotional campaign is based on back-to-school items, then the analyst may want to focus on rules that contain back-to-school items.

In this example, the rules table is illustrated in Figure 7-26 on page 168. The rules can be sorted by any column in this table, and various filtering capabilities

are available to enable the analyst to focus on a subset of item relationships of interest. When a taxonomy is specified in the associations model, rules relating entities across various levels of the taxonomy (for example, item → department) can be found.



Rule	▼ Support	Confidence	Lift	Al
[WWM WOVEN TOPS] ==> [MENS_WEAR]	2.4711%	40.2704%	0.8668	
[WTURTLENECKS] ==> [MENS_WEAR]	2.3705%	49.8414%	1.0728	
[WWM ACCESSORIES] ==> [WOMENS_WEAR]	2.0865%	60.2760%	1.6125	
[WSWEATERS] ==> [MENS_WEAR]	1.6968%	42.0299%	0.9046	
[MTURTLENECKS] ==> [WOMENS_WEAR]	1.6466%	40.0367%	1.0711	
[WWM ACCESSORIES] ==> [MENS_WEAR]	1.4077%	40.6681%	0.8753	
[MENS_WEAR]+[CHILDRENS_WEAR] ==> [WOMENS_WEAR]	1.0784%	50.7092%	1.3566	
[WOMENS_WEAR]+[CHILDRENS_WEAR] ==> [MENS_WEAR]	1.0784%	49.4810%	1.0650	
[WSLEEPWEAR] ==> [MENS_WEAR]	1.0784%	42.7717%	0.9206	
[MRUGBY] ==> [WOMENS_WEAR]	0.9729%	40.1452%	1.0740	
[MENS_WEAR]+[WWM ACCESSORIES] ==> [WOMENS_WEAR]	0.9377%	66.6071%	1.7819	
[WOMENS_WEAR]+[WWM ACCESSORIES] ==> [MENS_WEAR]	0.9377%	44.9398%	0.9673	
[0307385C1WM 50/50 COT/POLY T-NECK] ==> [MENS_WEAR]	0.8321%	52.3734%	1.1273	
[WACTIVE BOTTOMS] ==> [MENS_WEAR]	0.7843%	42.4490%	0.9137	
[MSLEEPWEAR] ==> [WOMENS_WEAR]	0.7516%	43.7135%	1.1694	
[WOMENS_WEAR]+[HOUSEWARES] ==> [MENS_WEAR]	0.6988%	51.4815%	1.1084	

Figure 7-26 Associations visualization example rules

The table of item sets used in constructing the rules is shown in Figure 7-27. This table is particularly useful for selecting a particular item set (containing one or more items) and displaying the rules and graph just for that item set. For example, if the analyst wants to focus on Women's Casual Wear (the highlighted item set in the table in Figure 7-27) for a promotion, then that item set can be selected and display only the rules containing Women's Casual Wear.

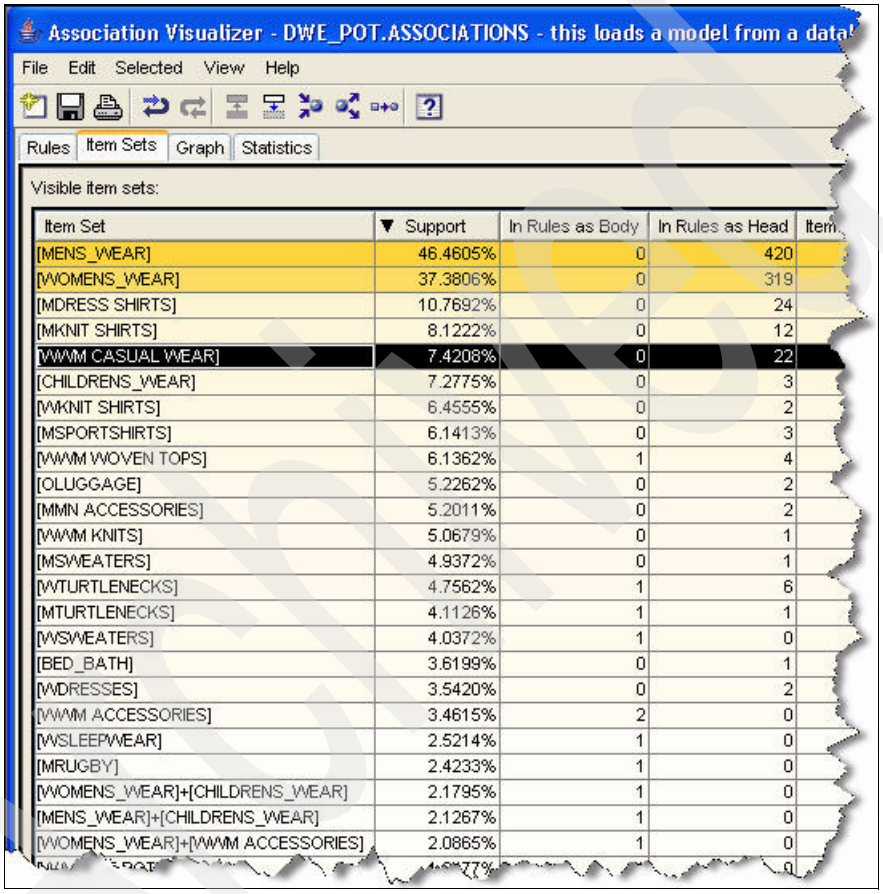


Figure 7-27 Associations visualization example: item sets

A graphical representation of the rules table is shown in Figure 7-28. As the figure clearly illustrates, displaying all the rules at once can make for a rather busy graph.

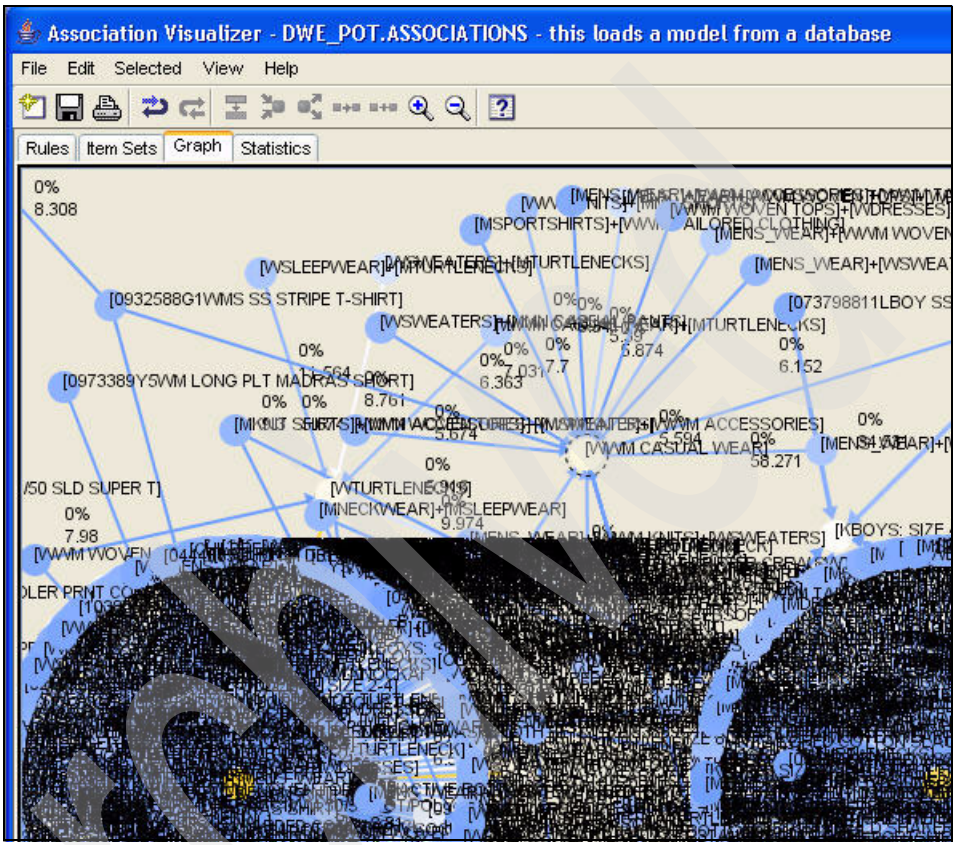


Figure 7-28 Associations visualization example: graph of all rules

But when a subset of rules of interest for a particular business objective is selected, the combination of the graph and rules table becomes much more informative. For example, if the objective is to identify a set of items associated with the Women's Casual Wear subdepartment for an upcoming promotion, then the Women's Casual Wear item set can be selected as explained for Figure 7-27 on page 169. Then only the rules containing Women's Casual Wear are displayed in the rules table and in the graph, as shown in Figure 7-29.

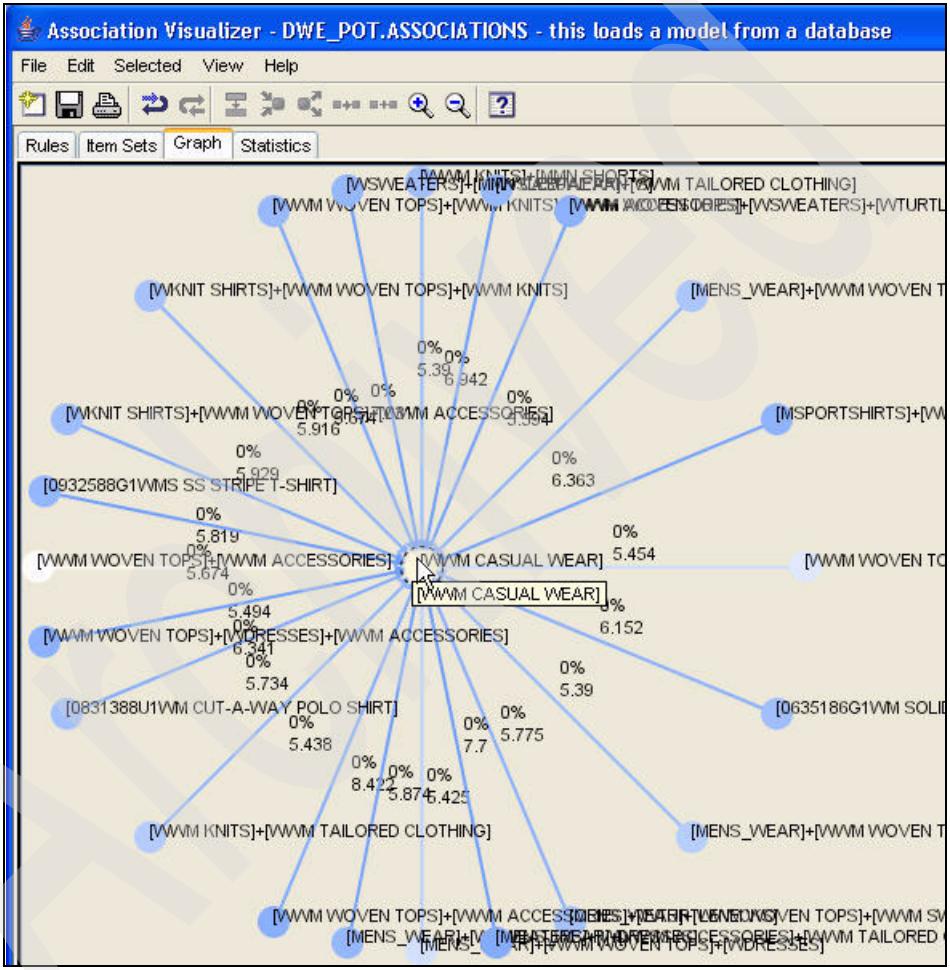


Figure 7-29 Associations visualization example: graph of rules for a selected item set

Once the relevant subset of rules has been found, the analyst can evaluate those rules in terms of the business objective. The rules involving Women's Casual Wear are shown in Figure 7-30. The first rule shows that when the item Solid D-Ring Belt is purchased, then an item in the Women's Casual Wear subdepartment will also be purchased in 45% of those transactions (per the Confidence column). Thus, this item might be put on sale to draw customers who will then also purchase related items in Women's Casual Wear.

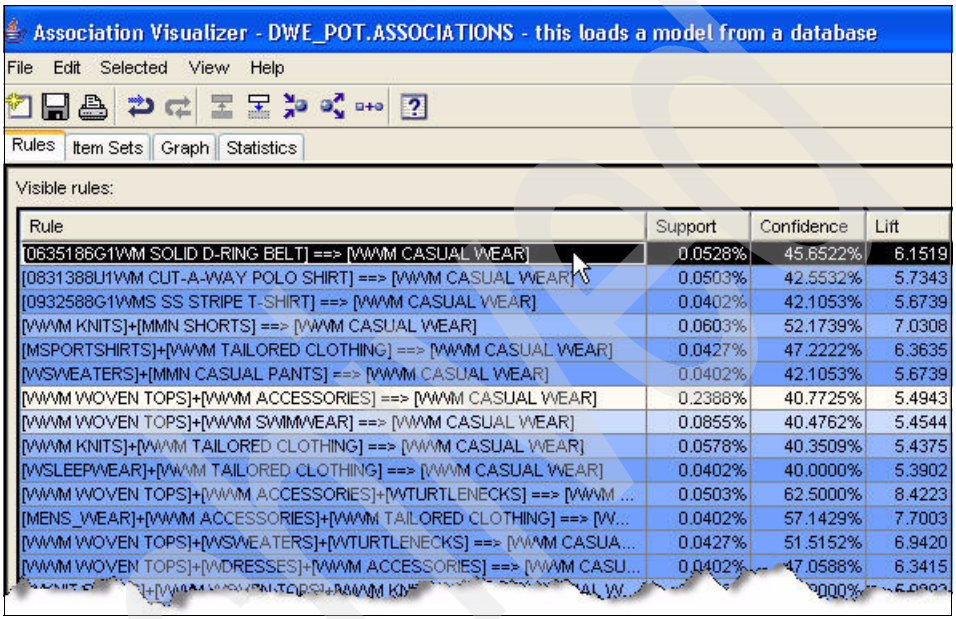


Figure 7-30 Associations visualization example: rules for a selected item set

7.3.3 Scoring with an associations rule model

Associations rule models can be applied to transactions in a scoring process to answer business questions such as:

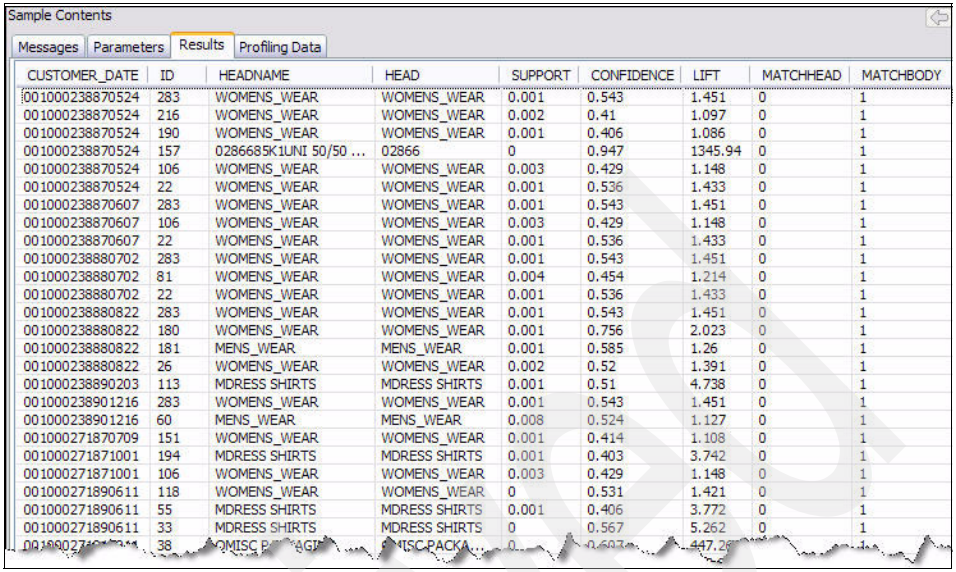
- Inventory management: Which items should be stocked up in preparation for promoting a particular item?
- Joint events:
  - Cross-selling or upselling: Given the items in a customer's basket, which other items is the customer most likely to buy?
  - Disease management: Given a set of symptoms and other attributes, which disease is a patient most likely to develop?
  - Failure mitigation: Given a part failure, what other failure is most likely also to occur?

As an example of real-time scoring, cross-selling recommendations can be computed in real time using an application that a customer service representative uses to check the items in a particular customer's market basket and then retrieve the associations rules that include those items in the rule bodies. The items in the heads of the rules are likely to be of interest to the customer because other customers have purchased these items together.

As an example of batch scoring, cross-selling recommendations can be computed by using an associations model to score a list of customers' transactions and then sorting each customer's transactions by the confidence values. A personalized mailing campaign can then be targeted at a set of customers who have purchased the body items but not the head items of relevant high-confidence rules.

During execution of the mining run, the scoring results are written out to a new table. A sample of the scoring results is shown in Figure 7-31 on page 174. The columns are interpreted as follows, where each row in the table represents a transaction:

- CUSTOMER_DATE: The transaction ID
- ID: The ID of the matching rule in the model (and the key to that rule in the output table of extracted rules)
- HEADNAME: Description of the item in the rule head
- HEAD: The ID of the item in the rule head
- SUPPORT: Support value of the matching rule
- CONFIDENCE: Confidence value of the matching rule
- LIFT: Lift value of the matching rule
- MATCHHEAD: Flag indicating whether or not the head item is included in the transaction (1=yes, 0=no)
- MATCHBODY: Number of matching body items or body item sets


Sample Contents

Messages | Parameters | Results | Profiling Data

CUSTOMER_DATE	ID	HEADNAME	HEAD	SUPPORT	CONFIDENCE	LIFT	MATCHHEAD	MATCHBODY
001000238870524	283	WOMENS_WEAR	WOMENS_WEAR	0.001	0.543	1.451	0	1
001000238870524	216	WOMENS_WEAR	WOMENS_WEAR	0.002	0.41	1.097	0	1
001000238870524	190	WOMENS_WEAR	WOMENS_WEAR	0.001	0.406	1.086	0	1
001000238870524	157	0286685K1UNI 50/50 ...	02866	0	0.947	1345.94	0	1
001000238870524	106	WOMENS_WEAR	WOMENS_WEAR	0.003	0.429	1.148	0	1
001000238870524	22	WOMENS_WEAR	WOMENS_WEAR	0.001	0.536	1.433	0	1
001000238870607	283	WOMENS_WEAR	WOMENS_WEAR	0.001	0.543	1.451	0	1
001000238870607	106	WOMENS_WEAR	WOMENS_WEAR	0.003	0.429	1.148	0	1
001000238870607	22	WOMENS_WEAR	WOMENS_WEAR	0.001	0.536	1.433	0	1
001000238880702	283	WOMENS_WEAR	WOMENS_WEAR	0.001	0.543	1.451	0	1
001000238880702	81	WOMENS_WEAR	WOMENS_WEAR	0.004	0.454	1.214	0	1
001000238880702	22	WOMENS_WEAR	WOMENS_WEAR	0.001	0.536	1.433	0	1
001000238880822	283	WOMENS_WEAR	WOMENS_WEAR	0.001	0.543	1.451	0	1
001000238880822	180	WOMENS_WEAR	WOMENS_WEAR	0.001	0.756	2.023	0	1
001000238880822	181	MENS_WEAR	MENS_WEAR	0.001	0.585	1.26	0	1
001000238880822	26	WOMENS_WEAR	WOMENS_WEAR	0.002	0.52	1.391	0	1
001000238890203	113	MDRESS SHIRTS	MDRESS SHIRTS	0.001	0.51	4.738	0	1
001000238901216	283	WOMENS_WEAR	WOMENS_WEAR	0.001	0.543	1.451	0	1
001000238901216	60	MENS_WEAR	MENS_WEAR	0.008	0.524	1.127	0	1
001000271870709	151	WOMENS_WEAR	WOMENS_WEAR	0.001	0.414	1.108	0	1
001000271871001	194	MDRESS SHIRTS	MDRESS SHIRTS	0.001	0.403	3.742	0	1
001000271871001	106	WOMENS_WEAR	WOMENS_WEAR	0.003	0.429	1.148	0	1
001000271890611	118	WOMENS_WEAR	WOMENS_WEAR	0	0.531	1.421	0	1
001000271890611	55	MDRESS SHIRTS	MDRESS SHIRTS	0.001	0.406	3.772	0	1
001000271890611	33	MDRESS SHIRTS	MDRESS SHIRTS	0	0.567	5.262	0	1
001000271890611	38	OMISC P AGI	MISC-PACKA...	0	0.607	447.26	0	1

Figure 7-31 Associations scoring results: sampled contents of output table

In Figure 7-31, a particular transaction may be represented by more than one row. This is because the item(s) in the transaction may appear in more than one rule. Hence, a customer having a basket containing certain item(s) may be likely to purchase more than one additional item and therefore could be targeted for multiple promotional offers.

An example is illustrated in Figure 7-32. In Table 1, a particular transaction (where CUSTOMER_DATE is the transaction ID) has been scored using two rules with IDs of 71 and 297, which are shown in Table 2. This transaction was scored with two rules because it contained some of the items in both of these rules. The MATCHHEAD value of 0 indicates that the customer who made this transaction did not purchase either of the head items of the two rules. Thus, this customer may be a good candidate for an offer for either or both of the head items in rules 71 and 297. When implemented appropriately, this finding could be applied either in real time while the customer is in the store making a purchase or in batch mode when planning a promotional campaign.

Table 1. Scored Transaction with Two Rule Matches

CUSTOMER_DATE	ID	HEADNAME	HEAD	SUPPORT	CONFI-DENCE	LIFT	MATCH HEAD	MATCH BODY
001124525910906	71	1240090B2PIMA HAND TOWEL	12400	0.1986%	79.0%	259.7	0	2
001124525910906	297	0953589G1CARDED COTTON HAND TOWEL	9535	0.1936%	77.8%	300.4	0	2

Table 2. Matching Rules from Associations Model

ID	BODYID	HEAD	HEADNAME	LENGTH	BODYTEXT	SUPPORT	CONFI-DENCE	LIFT
71	1611	12400	1240090B2PIMA HAND TOWEL	3	[1240390B2PIMA SLD BATH TOWEL] [1240190B2PIMA SLD WASHCLOTH SET/2]	0.1986%	79.0%	259.7
297	1222	09535	0953589G1CARDED COTTON HAND TOWEL	3	[0953789G1CARDED COTTON BATH TOWEL] [0953689G1CARDED COT WASHCLOTH SET/2]	0.1936%	77.8%	300.4

Figure 7-32 Associations scoring results example

7.3.4 Sampling transactions

Although the Table Source operator allows a sampling rate between 0 and 100 percent, sampling should not be done this way for associations mining. Because a transactions table contains multiple records per transaction, selecting a random sample of the records would not preserve the character of the transactions and therefore would not lead to valid associations rules. If a sample is needed for mining, then a table or view containing a list of the unique transaction IDs in the transactions table should be created. A random sample of transaction IDs is selected from this list. The sample of transaction IDs is then joined to the original transactions table to extract all of the records for each transaction in the random sample.

An example of this sampling process is illustrated in Figure 7-33. The Table Source (1) contains transactions consisting of items purchased in those transactions. In this example, TRANSACTION_DATE is the transaction identifier, and ITEM_ID is the item identifier. From the table source, the Distinct operator (2) selects a list of the unique transaction IDs. The Sampler operator (3) takes a random sample of the desired size (10 percent in this case) from the list of unique transaction IDs in (2). In the Table Join operator (4), the randomly sampled transaction IDs are joined with the source transactions table (1). This joining process extracts all of the records for the random sample of transaction IDs, producing a proper random sample for associations mining. This sample of transactions then flows from (4) to the Input port of the Associations operator (not shown here).

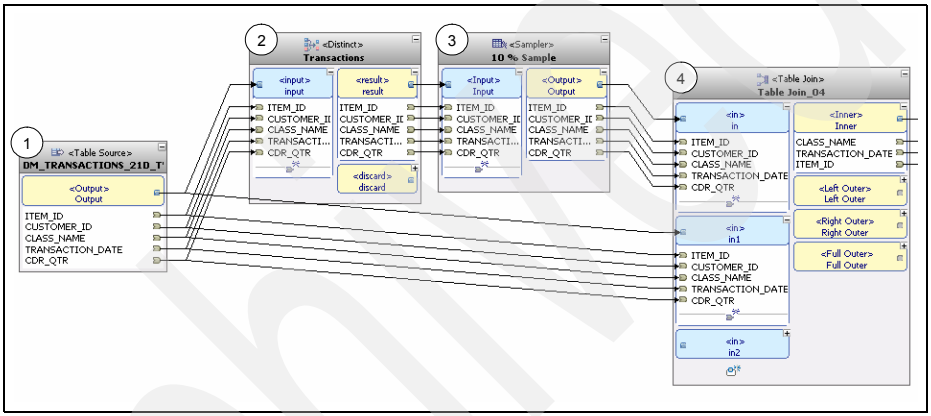


Figure 7-33 Randomly selecting transactions

7.4 Sequences

The following example illustrates the sequences method. In this example, we continue the market basket analysis scenario for the associations analysis. Here, instead of looking for item affinities within a given transaction, we want to find the next most likely purchase when a customer buys a particular item. A sequence (or transaction group) is a time-ordered set of item sets, where an item set (that is, a transaction) is a collection of one or more individual items. By looking at sequences of items purchased across multiple transactions over time, we can find relationships useful for promotions and for cross-selling or upselling offers.

7.4.1 Building a sequences mining flow

For sequences analysis, we need a transactions table containing the two columns required for associations, that is, the transaction identifier for each transaction and the items contained in that transaction, plus a third column representing the transaction group. In this example, the transaction group is represented by the customer ID. As with associations, we also need tables containing item descriptions, taxonomy structure, and descriptions of the elements of the taxonomy.

The mining flow for sequences modeling is shown in Figure 7-34 on page 178. The mining flow is constructed as follows:

- As usual, the flow begins with an input Table Source operator (1) and a Select List operator (2) to provide the input columns.
- The selected columns flow into the data Input port of a Sequences operator (8) where parameters and other settings for building the mining modeling are specified.
- Since this model employs a taxonomy with one level above the product level (Product < Group), another Table Source operator (3) provides a name mapping of group IDs to group descriptions.
- Another Table Source operator (4) and Select List operator (5) provide a name mapping for the product ID as well as the product-group taxonomy, given the content of the source table in this case.
- The mining model created according to the specifications and parameters in the Sequences operator (8) flows to a Visualizer operator (9) that displays the model results graphically.
- The Scorer operator (10) reads in the model from (8) and applies it to new transactions from (6) and (7). The scored results are written to a new table defined by a Table Target operator (11) using the Create Suitable Table function.
- The Sequences Extractor operator (12) extracts sequence rules from the model and writes them out to a Table Target operator (13) created using the Create Suitable Table function.

*

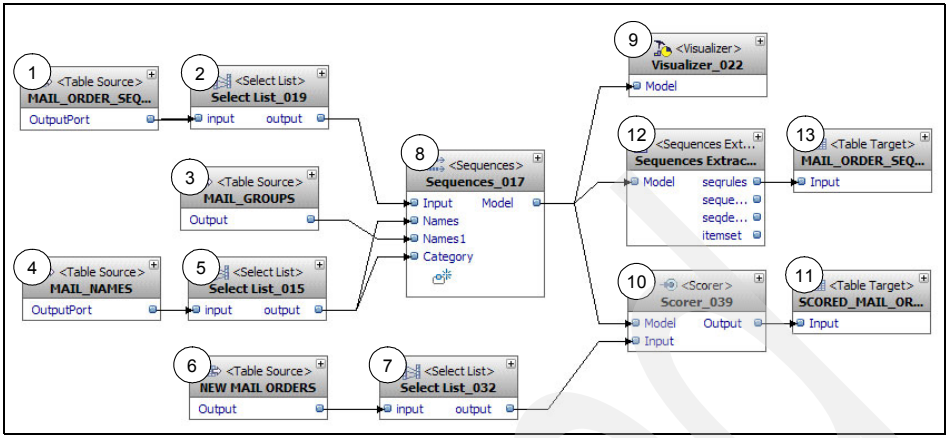


Figure 7-34 Sequences mining flow

Although the Table Source operator allows a sampling rate between 0 and 100 percent, sampling should not be done this way for sequences mining. Because a transactions table contains multiple records per transaction as well as multiple transactions per transaction group (that is, customer in this case), selecting a random sample of the records would not preserve the character of the transaction groups and their respective transactions and therefore would not lead to valid sequences rules. For an example of how to sample transaction groups properly, see 7.4.4, “Sampling transaction groups” on page 185.

The mining settings in the Sequences operator (8) are shown in Figure 7-35 on page 179. Parameters are specified as follows:

- As for associations, the group column refers to the input column representing the transaction ID, which may be a time stamp.
- The sequence column refers to the input column representing the transaction group ID (for example, customer ID).
- The maximum rule length parameter sets the maximum number of items that a sequences rule may contain. This parameter must be greater than or equal to 2. For example, a maximum rule length of 4 means one item in the head (always) and up to three items in the body of a rule.
- The maximum number of rules parameter sets the maximum number of sequences rules that the model may contain. The default value of 0 means that no maximum is set.
- The minimum confidence parameter, expressed as a percentage from 0 to 100, sets the minimum confidence that a rule must have to be in the model.
- The minimum support parameter, expressed as a percentage from 0 to 100, sets the minimum support that a rule must have to be in the model.

- The number of bins specifies the number of buckets (bins) automatically created for numerical data used as items. The minimum value is 2 with no upper limit and a default value of 5.
- Filters and other constraints can be specified as optional parameters, although sequences rules can also be filtered after the model has been created.

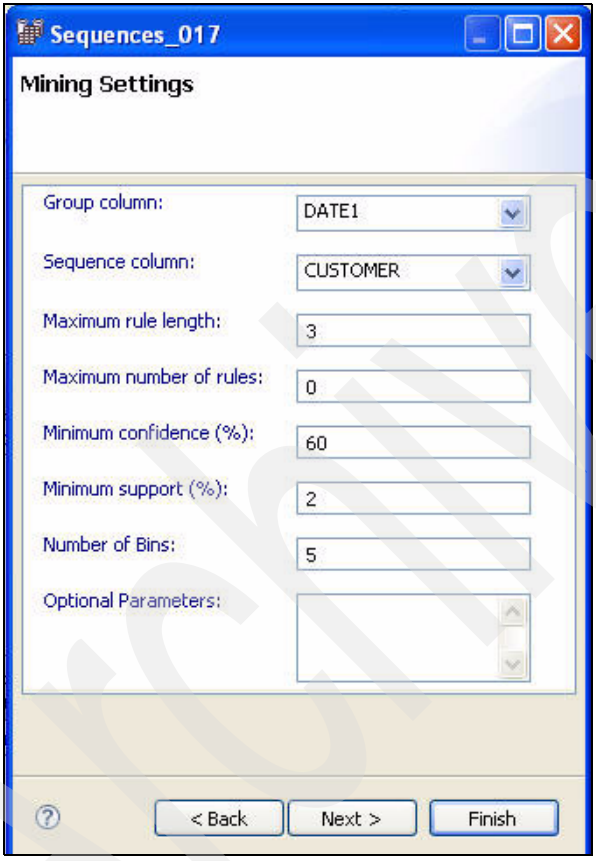


Figure 7-35 Sequences mining flow: specifying mining settings

Name maps, taxonomy, and column properties in the Sequences operator are defined similarly for sequences as they are for associations. The mining flow shown in Figure 7-34 on page 178 is executed by selecting **Mining Flow** → **Execute** from the Design Studio menu bar. Upon completion, a visualization of the model generated by the mining flow is automatically displayed.

7.4.2 Interpreting a sequences rule model

The sequences visualizer presents a set of rules describing series of items within transactions that occur sequentially over time. The sequences visualization includes tables of sequence rules, item sets on which the rules are based, and summary statistics about the rules. In this example, the sequence rules are depicted in Figure 7-36. This figure has a significant amount of information contained in it and is shown only as an example. It is not required for you to be able to read all of that information.

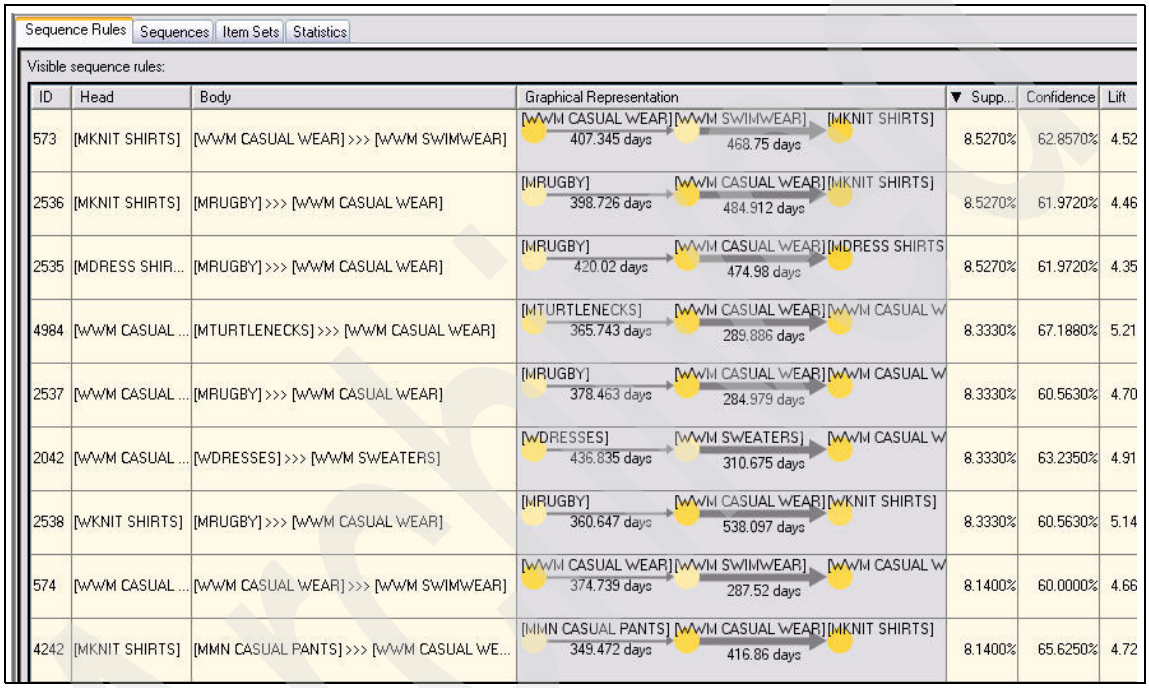


Figure 7-36 Sequences visualization example: sequence rules

The columns of the sequence rules table in Figure 7-36 list the sequence rule ID, the head item in the sequence, the items in the body transaction(s) in the sequence, a graphical representation of the sequence rule including the mean time between transactions in the rule, and the support, confidence, and lift of the sequence rule. The mean number of days between transactions is calculated from the transaction ID and is meaningful when a time stamp is used as the ID. The support is calculated as the number of transaction groups containing the sequence divided by the total number of transaction groups. The confidence is calculated as the number of transaction groups containing the sequence divided by the number of transaction groups containing the body of the sequence. The lift is calculated as the number of transaction groups containing the sequence

divided by the number of transaction groups expected to contain the sequence (given the supports of all item sets in the sequence and the assumption that the occurrences of all item sets are statistically independent).

To illustrate, consider the first sequence rule, Rule 573, in Figure 7-38 on page 184. In this sequence, a purchase of any item in the Women's Casual Wear group [WWM CASUAL WEAR] is followed by a purchase of an item in the Women's Swim wear group [WWM SWIMWEAR] and then by an item in the Men's Knit Shirts group [MKNIT SHIRTS]. The mean time between the first and second purchases is 407.3 days, with an average of 468.7 days between the second and third. The support is 8.5%, indicating that this sequence occurs in 8.5% of all transaction groups considered in this analysis. The confidence indicates that customers tend to purchase an item in Men's Knit Shirts 62% of the time after purchasing items in Women's Casual Wear and then Women's Swim wear. The lift of 4.52 indicates a reasonably strong (non-random) relationship. This relationship might be used in designing a seasonal campaign or targeted promotion, while the relatively long time between purchases might suggest the need for an incentive to encourage customers to shop more frequently at this retail chain.

Summary statistics for the sequences analysis are presented in Figure 7-37 on page 182. The number of *visible sequence rules* refers to the number of rules displayed in the Sequence Rules table, which can change as rules or item sets are filtered interactively. Other statistics are computed as follows:

- The number of transactions (9,593) is equal to the number of distinct transaction IDs in the transactions table, which is operator (1) in Figure 7-34 on page 178.
- The average number of items per transaction (2.58) is equal to the number of rows (24,749 with one row per item) in the transactions table divided by the number of distinct transaction IDs in the transactions table.
- The number of transactions groups (516) is equal to the number of distinct customer IDs in the transactions table.
- The average number of transactions per transaction group (18.59) is equal to the number of distinct transaction IDs divided by the number of distinct customer IDs in the transactions table.
- The number of singleton item sets (187) is the number of item sets that contain only one item. Although singleton item sets do not contribute to finding associations within item sets (transactions), they may be useful in constructing sequences across item sets.
- The number of item sets used in sequence rules (30) is the total number of item sets that occur within the complete set of sequence rules, given the threshold parameters for support and confidence.

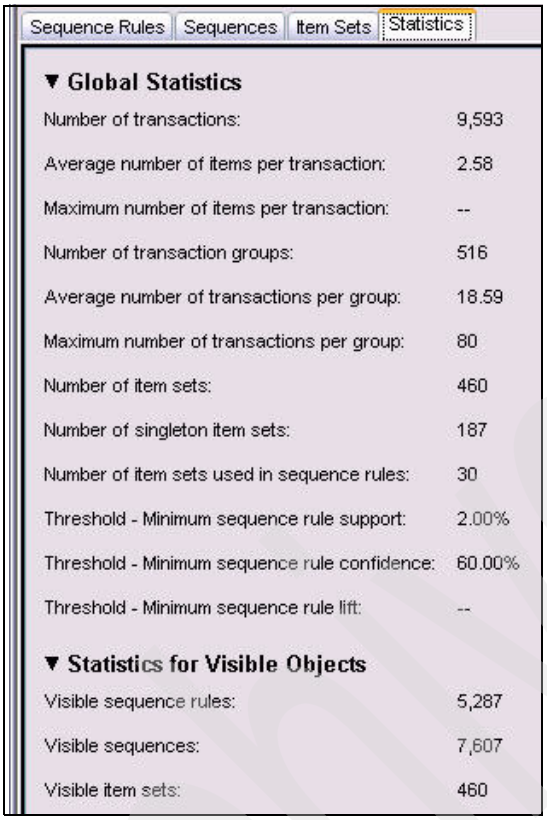


Figure 7-37 Sequences visualization example: summary statistics

7.4.3 Scoring with a sequences rule model

Sequences rule models can be applied in a scoring process to answer such business questions as:

- Upselling: Given a purchases or series of purchases, what is the next most likely purchase that a customer will make?
- Disease management: Given a series of symptoms and episodes, what is the next most likely condition that a patient will develop?
- Failure mitigation: Given previous problems, what is the most likely type of system or part failure that will occur next?

As an example of real-time scoring, cross-selling or upselling recommendations can be computed in real time using an application to check the items in a particular customer's transactions history and then retrieve the sequences rules that include those items in the rule bodies. The items in the heads of the rules are likely to be purchased next because other customers have purchased these items sequentially. A customer service representative can then make an offer to induce the customer to make the next purchase now instead of later (and possibly from a competitor).

As an example of batch scoring, cross-selling or upselling recommendations can be computed by using a sequences model to score a list of customers' transactions histories and then sorting each customer's scored rules by the confidence values. A personalized mailing campaign can then be targeted at a set of customers who have purchased the body items but not the head items of relevant high-confidence rules.

During the execution of the mining run, the scoring results are written out to a new table. A sample of the scoring results is shown in Figure 7-38 on page 184. The columns are interpreted as follows, where each row in the table represents a transaction:

- CUSTOMER: Transaction group ID (customer ID in this case).
- ID: The ID of the matching rule (and the key to that rule in the output table of extracted rules).
- HEADNAME: Description of the item in the matching rule head.
- HEAD: Head item set of the matching rule.
- SUPPORT: Support value of the matching rule.
- CONFIDENCE: Confidence value of the matching rule.
- LIFT: Lift value of the matching rule.
- MATCHHEAD: If the head item is contained in the item set or if the head item set is part of the matching sequence, then the match head value is 1; if the head item is not contained in the item set or if the head item set is not part of the matching sequence, then the match head value is 0.
- MATCHBODY: Number of matching body items or body item sets.

CUSTOMER	ID	HEADNAME	HEAD	SUPPORT	CONFIDENCE	LIFT	MATCHHEAD	MATCHBODY
001000728	3799	MDRESS SHIRTS	0	0.037	0.826	4.35	0	1
001000728	3800	MKNIT SHIRTS	36	0.037	0.826	4.456	0	1
001000728	3801	WWM WOVEN TOPS	95	0.029	0.652	4.054	0	1
001000728	3802	MSPORTSHIRTS	138	0.029	0.652	3.959	0	1
001000728	4132	MDRESS SHIRTS	0	0.043	0.688	3.62	0	1
001000728	4981	MDRESS SHIRTS	0	0.021	0.688	3.62	0	1
001000728	5283	MDRESS SHIRTS	0	0.035	0.857	4.513	0	1
001000728	5284	MSPORTSHIRTS	138	0.027	0.667	4.047	0	1
001000728	5285	OLUGGAGE	163	0.025	0.619	3.729	0	1
001001191	4132	MDRESS SHIRTS	0	0.043	0.688	3.62	0	1
001001191	4314	MDRESS SHIRTS	0	0.091	0.61	3.214	0	1
001001191	4315	MDRESS SHIRTS	0	0.025	0.812	4.278	0	1
001001191	4317	MKNIT SHIRTS	36	0.021	0.688	3.708	0	1
001001191	4335	MDRESS SHIRTS	0	0.021	0.733	5.148	0	2
001001191	4584	MDRESS SHIRTS	0	0.021	0.611	4.29	0	2
001001191	4821	MKNIT SHIRTS	36	0.045	0.622	3.353	0	1
001001191	4869	MDRESS SHIRTS	0	0.027	0.824	4.336	0	1
001001191	4870	MKNIT SHIRTS	36	0.027	0.824	4.442	0	1
001001191	4871	MSPORTSHIRTS	138	0.023	0.706	4.285	0	1
001001191	4872	MMN TAILORED C...	172	0.023	0.706	4.99	0	1
001001191	4873	WWM KNITS	188	0.023	0.706	4.9	0	1
001001191	4874	WTURTLENECKS	455	0.023	0.706	4.553	0	1
001001191	4979	MKNIT SHIRTS	36	0.031	0.615	3.319	0	1
001001191	5175	MDRESS SHIRTS	0	0.031	0.762	4.012	0	1
001001191	5176	WWM CASUAL W...	66	0.025	0.619	3.603	0	1
001001	5177	WKNIT SHI	119	0.031	0.762	4.854		

Figure 7-38 Sequences scoring results: sampled contents of output table

A particular customer may be represented by more than one row in the scoring output table (Figure 7-38) because his body items or body item sets may match more than one sequences rule. Hence, a customer having a sequence containing certain item(s) may be likely to purchase more than one additional item and therefore could be targeted for multiple promotional offers, either in real-time while he is in the store or through a promotional campaign.

An example from the table of extracted sequence rules, represented by operator (13) in Figure 7-34 on page 178, is shown in Figure 7-35 on page 179. Rule 3800 matches the body item for Customer 001000728 in the second row of Figure 7-38. According to Rule 3800, this customer is likely to purchase an item in Men's Knit Shirts, with a confidence of 82.6%, as depicted in Figure 7-39.

ID	BODY ID	HEAD	BODYTEXT	HEADNAME	LENGTH	# ITEMS	SUPPORT	CONFI-DENCE	LIFT	MEAN DAYS	STD (MEAN DAYS)
3800	7004	37	[0234885A1MN R SILK FOULARD TIE]	[MKNIT SHIRTS]	2	2	3.7%	82.6%	4.456	564.5	469.0

Figure 7-39 Sequences rule example

7.4.4 Sampling transaction groups

Although the Table Source operator allows a sampling rate between 0 and 100 percent, sampling should not be done this way for sequences mining. Because a transactions table contains multiple records per transaction as well as multiple transactions per customer, selecting a random sample of the records would not preserve the character of the transaction group and their respective transactions and therefore would not lead to valid sequences rules. If a sample is needed for mining, then a table or view containing a list of the unique transaction group IDs (for example, customers) in the transactions table should be created. A random sample of transaction group IDs is selected from this list. The sample of transaction groups is then joined to the original transactions table to extract all of the transactions records for each transaction group ID in the random sample.

An example of this sampling process is illustrated in Figure 7-40 on page 186. The Table Source (1) contains customer-related data including the customer identifier called IDV_IP_ID. From this customer data source, the Distinct operator (2) selects a list of unique customer IDs represented by the column IDV_IP_ID. Here, the customer ID represents the transaction group ID. The Sampler operator (3) takes a random sample of the desired size (10 percent in this case) from the list of unique transaction group IDs in (2). In the Table Join operator (4), the randomly sampled transaction group IDs are joined with the source transactions table (5), which contains the transaction group ID (which is CUSTOMER_ID in this source table, corresponding one-to-one with IDV_IP_ID in the other source table), the transaction identifier (TRANSACTION_DATE) for each customer's transactions, and the item identifier (ITEM_ID) for the items purchased in each transaction.

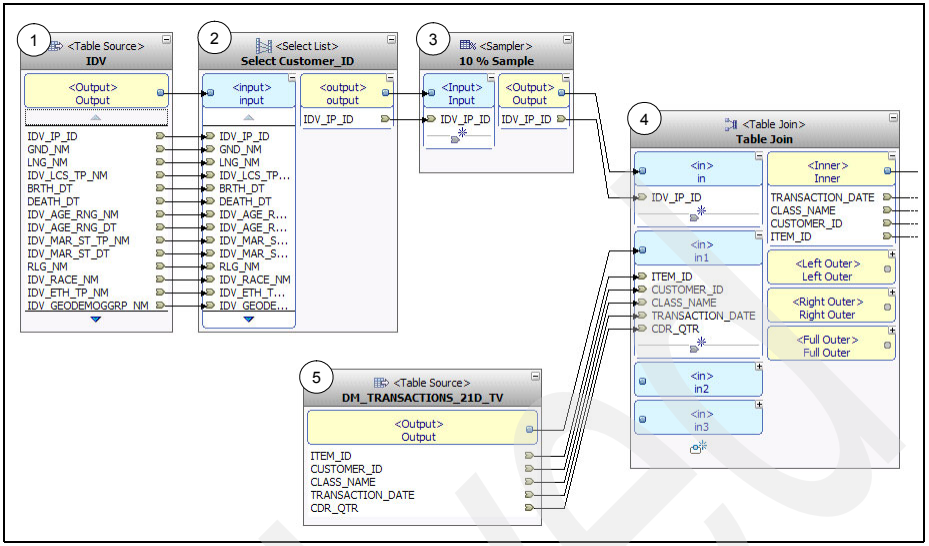


Figure 7-40 Randomly selecting sequences

The table join condition for (4) is shown in Figure 7-41. This joining process extracts all of the records for the random sample of transaction groups, producing a proper random sample for sequences mining. This sample then flows from (4) to the input port of the Sequences operator (not shown in Figure 7-40).

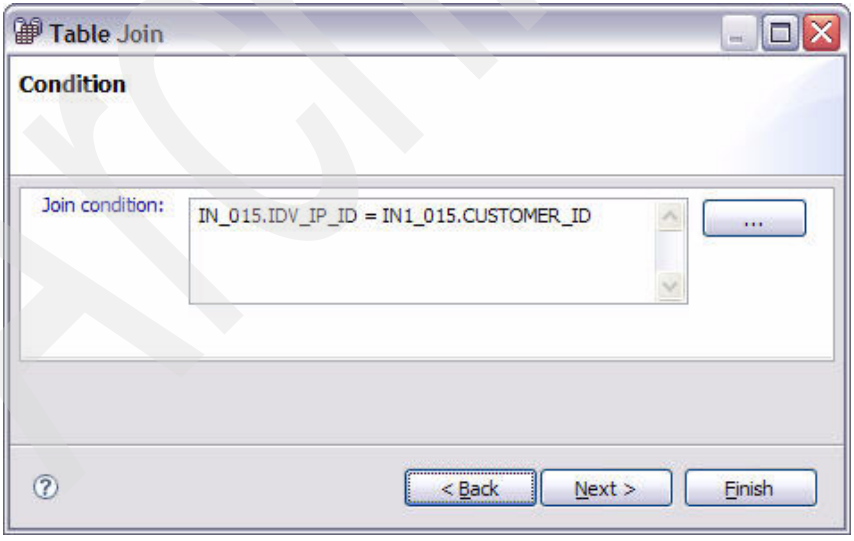


Figure 7-41 Randomly selecting sequences: table join condition

7.5 Classification

The following example illustrates the classification method. The business problem is to reduce auto insurance customer attrition by identifying those customers at high risk of attriting and targeting them with retention incentives. Specifically, we want to predict whether a customer will voluntarily close his auto insurance account (attrit).

For a predictive model, we need variables representing the characteristics of customers and their accounts. We also need a response variable (called a *dependent variable*) for the behavior of interest, which in this case is whether or not a customer has attrited. Here, the dependent variable is categorical rather than numeric, so a classification method is appropriate. Using an input table containing historical data on current and past customers, including whether or not they attrited as of some point in time, we can build a model to predict whether or not a customer will attrit in the future. In this case, the dependent variable ATTRIT has possible values of Y (yes) and N (no).

7.5.1 Building a classification mining flow

The mining flow is shown in Figure 7-42. As usual, the flow begins with an input Table Source operator (1) and Select List operator (2). The table source contains one record per customer with columns representing information about each customer's coverage, vehicles, and premiums, as well as attributes of each customer's insurance agent. Each record also includes that customer's attrition response (Y or N).

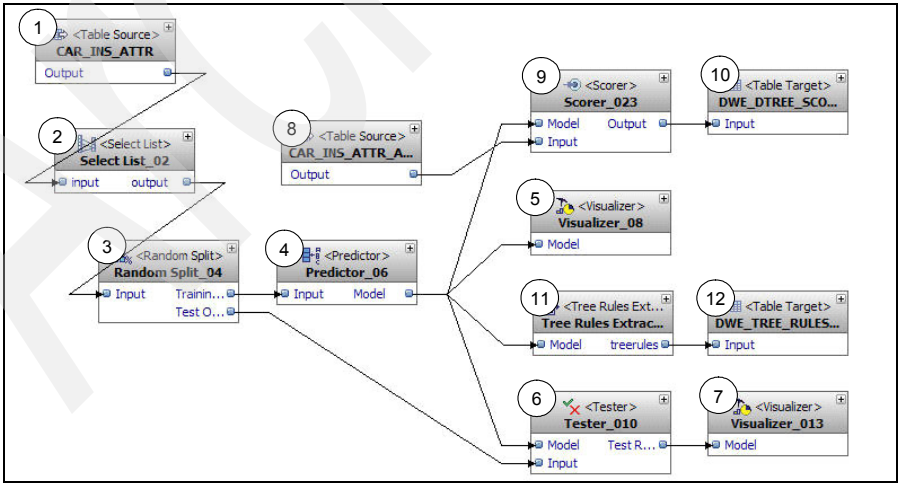


Figure 7-42 Mining flow for decision tree classification

In Figure 7-42 on page 187, a Random Split operator (3) divides the input data into two randomly selected subsets, one for training (building) the model and the other for testing (validating) the model to assess its expected performance when applied to new customers, as illustrated in Figure 7-43 on page 189.

Note: Unsupervised learning techniques such as clustering, associations, and sequences do not employ training and testing sets. See Chapter 4, “Data mining methods” on page 39 for more discussion about training and testing sets.

The Random Split operator allows either simple random sampling (the default “no stratified sampling”) or stratified sampling using one of the columns in the input table. Selecting a column will create a stratified sample for the training data set. This means that, based on the complement of the testing data set, a sample is created whereby the values of the stratified sampling column occur with approximately the same frequency, which may be useful when the distribution of the values of the target column (dependent variable) is unbalanced, for example, 99 percent NO and 1 percent YES.

Note: This is a form of *oversampling* or *enriched sampling* and is not the same as proportional stratified random sampling in which a sample is drawn for each value or stratum in proportion to its occurrence in the source data. See 7.7, “Oversampling to create an enriched training set” on page 206 for more discussion of oversampling.

Although using a stratified sample may improve the trained model in some cases, the confidence values computed by the classification model are not correct on a data set that does not have the same characteristics as the stratified sample (for example, the testing data set). Using stratified sampling requires statistical expertise, so the default (no stratified sampling) is recommended for those who do not have such knowledge.

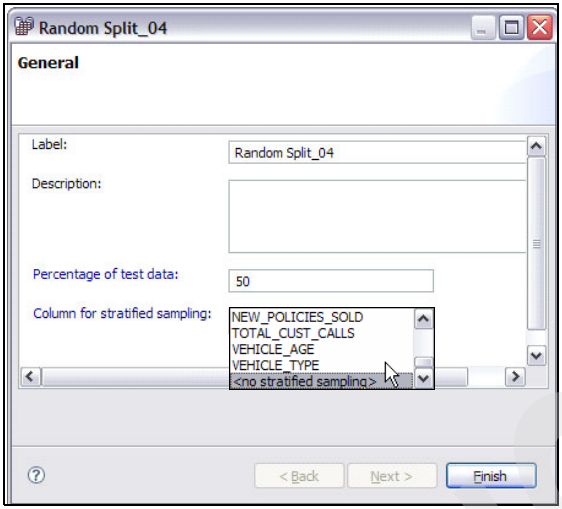


Figure 7-43 Random Split operator

In Figure 7-42 on page 187, the training data flows from the Random Split operator (3) to the Predictor operator (4), which executes the appropriate predictive mining algorithm (classification if the dependent variable is categorical or regression if it is numeric) and sends the trained model to a Visualizer operator (5). The testing data from (3) and the model from (4) flow to the Tester operator (6), which applies the model to the testing data and passes the test results to another Visualizer operator (7). Upon execution of the mining flow, then, two visualizations will be displayed to show the training and testing results. The testing results provide the appropriate indication of the accuracy that can be expected when the model is applied to new data.

The Predictor operator (4) offers the choice of three classification algorithms: decision tree, logistic regression, and Naïve Bayes. The decision tree is appropriate when the class (dependent) variable contains two or more values (for example, Yes, No; Red, Blue, Green), and its visualization produces the most information for model interpretation. Logistic regression is useful when the class variable represents a binary outcome (for example, Yes, No). Naïve Bayes Bayes is a probabilistic classifier based on a strong (“naive”) independence assumption about the variables in the model, and it requires only a small amount of data compared to other classification methods. It is particularly useful when textual data has been structured (using a text mining technique) and combined with the other structured data in the source table. Default parameter settings are recommended for all of the classification methods. The decision tree is used in this example to illustrate the variety of training and testing results.

For scoring, the model from (4) and a Table Source operator (8) containing new records are passed to a Scorer operator (9), which applies the model to the input records to generate the scoring information and then write the results to an output table specified in a Table Target operator (10) in Figure 7-42 on page 187. The scoring results include the predicted class value (Y or N) and the confidence of that prediction for each input record.

In Figure 7-42 on page 187, a Tree Rules Extractor operator (11) extracts the decision tree rules and writes them out to a table defined in a Table Target operator (12). This extractor is meaningful only for the decision tree classifier, not logistic regression or Naive Bayes classification.

7.5.2 Interpreting a classification model

The classification visualizer presents information to evaluate a decision tree classification model. In this example of predicting whether a customer will volunta rily close his account (attrit), the dependent variable (that is, the outcome, response, or class variable) is ATTRIT with possible values of Y (yes) and N (no). Given the business question, we are particularly interested in the Y outcome.

The visualization of the decision tree model shows several views for interpreting the trained model. The *confusion matrix* for the training set, shown in Figure 7-44, reports the counts of correct and incorrect predictions by ATTRIT classes (Y, N). The overall number of correct predictions is 81 percent, that is, 1,307 out of 1,621 customers in the training set. For the customers who actually attrited (Y), the model correctly predicted 427 out of 601, or 71 percent. Thus, the model is less accurate for the class of primary interest (Y) than the overall proportion of correct predictions suggests.

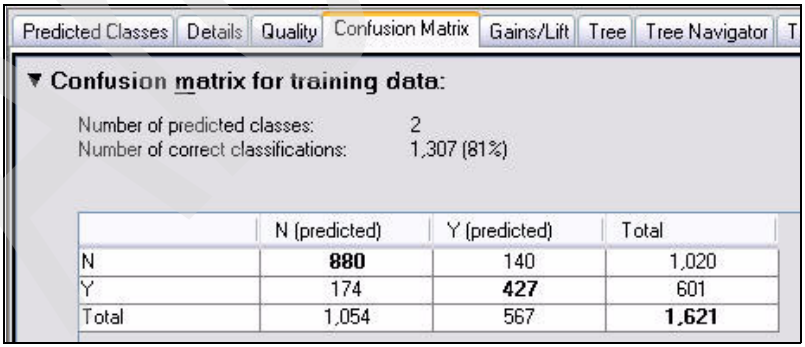


Figure 7-44 Decision tree classification example: confusion matrix for training set

Similarly, a confusion matrix is computed for the testing set. As shown in Figure 7-45, the overall percentage of correct classifications is 73 percent, and the percentage of correct predictions for attriters (Y) is 60 percent (328 out of 546). The lower overall accuracy typically is less with the testing set because that data was not used in building the model. Thus, the testing result more realistically indicates how well the model will perform when applied to new customer data, assuming that the new data is similar to the testing data.

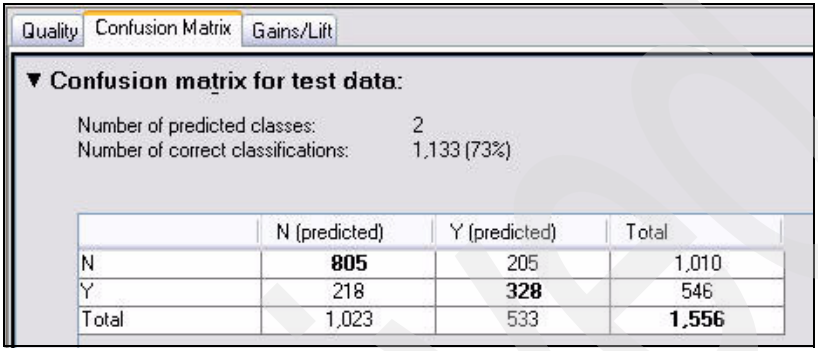


Figure 7-45 Decision tree classification example: confusion matrix for testing set

Model quality metrics are provided in the visualizations of both the trained model and the testing results, as illustrated in Figure 7-46. The quality based on the testing results is more relevant for how well the model can be expected to perform when applied to score new customers or rescore existing customers.

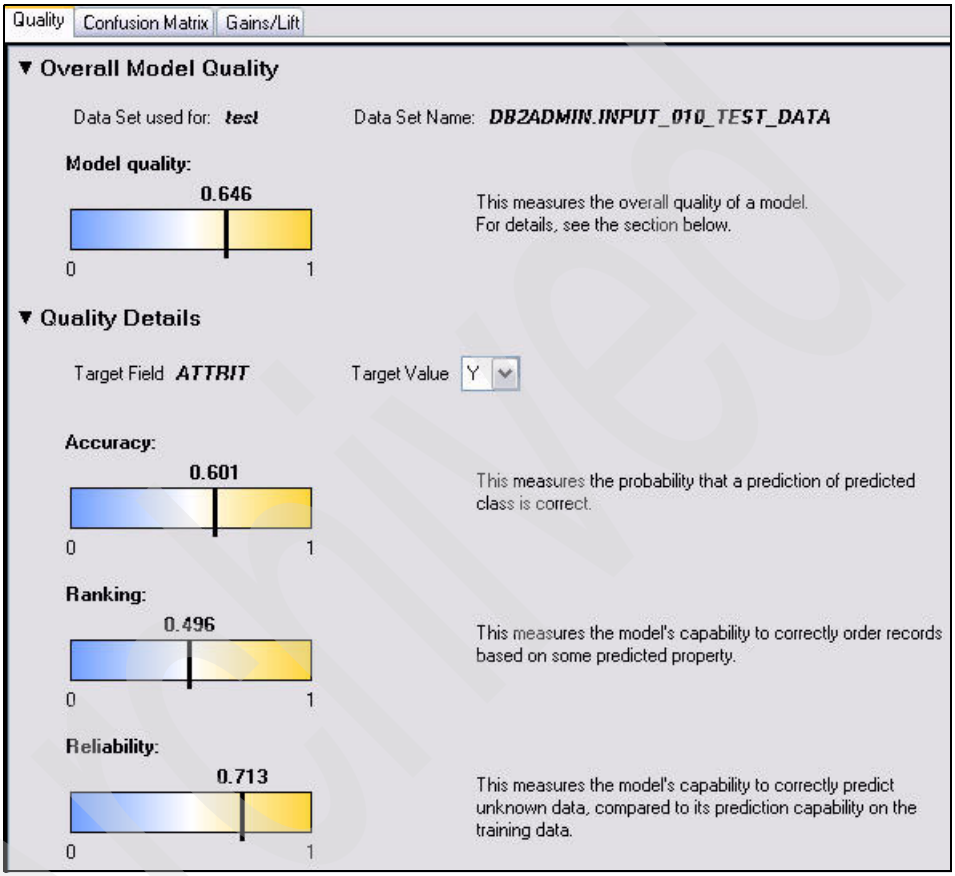


Figure 7-46 Decision tree classification example: model quality

In Figure 7-46, the *overall model quality* compares the predictive performance of a model to a trivial model that always returns the mean of the target field (ATTRIT in this case) as the prediction. An overall model quality value of 0 indicates that the predictive performance of the model is no better than predicting the mean, while a value close to 1 indicates that the predictive performance of the model is far superior to predicting the mean. A caveat is that, while we want the prediction quality to be as good as possible, a model quality that appears “too good” (for example, very close to 1) may indicate the presence of a causal dependency between the target field and one or more input fields, meaning that the

dependent variable is highly correlated with one explanatory variable or a linear combination of explanatory variables. (The field importance chart is helpful in understanding this.) For example, we may want to predict the amount of claims expected for a new insurance customer. A high-quality model based on existing customer data might reveal that customers with many accidents also have a large amount of claims. This model is not useful, however, because the number of claims is not known at the time when the prediction must be made.

Accuracy is the probability that a prediction is correct. It ranges from 0 (low) to 1 (high). In this case, the predicted value of interest is Y, for which the accuracy value is 0.601 (328 correctly predicted Y out of 546 total Y, from Figure 7-45 on page 191).

Ranking is a measure of the model's ability to order records correctly, calculated based on the order of the testing data records when sorted by the predicted values with the order of the same data records when sorted by the actual values of the target fields. In terms of a gains curve, the ranking value is the ratio of the area under the model curve to the area under the optimal or perfect model curve. The ranking value ranges from -1 to 1, where a value close to 1 indicates that the model ranks the records very well, while a negative value indicates that the model is worse than random and may suggest a data problem.

Reliability is a measure of the model's ability to predict unknown data. It ranges from 0 (low) to 1 (high) and is computed as the ratio of the error rate for the training set to the error rate for the testing set. From Figure 7-44 on page 190 and Figure 7-45 on page 191, reliability is calculated as ((1,621-1,307)/1,621) / ((1,556-1,133)/1,556) = 0.713 as shown in Figure 7-46 on page 192.

In the split-screen view in Figure 7-47, the decision tree model is shown on the left. On the right, the decision path and details for the highlighted tree node (Node 1.2.1.1.) is displayed. An individual whose attributes result in his being classified into this node receives a score of the dominant value (Y in this node) with the confidence indicated by the purity of the score (21 out of 25, or 84 percent Y). Thus, when this model is used for scoring new customers (or for rescoring existing customers), anyone classified into this node is considered to have an 84 percent likelihood of attriting.

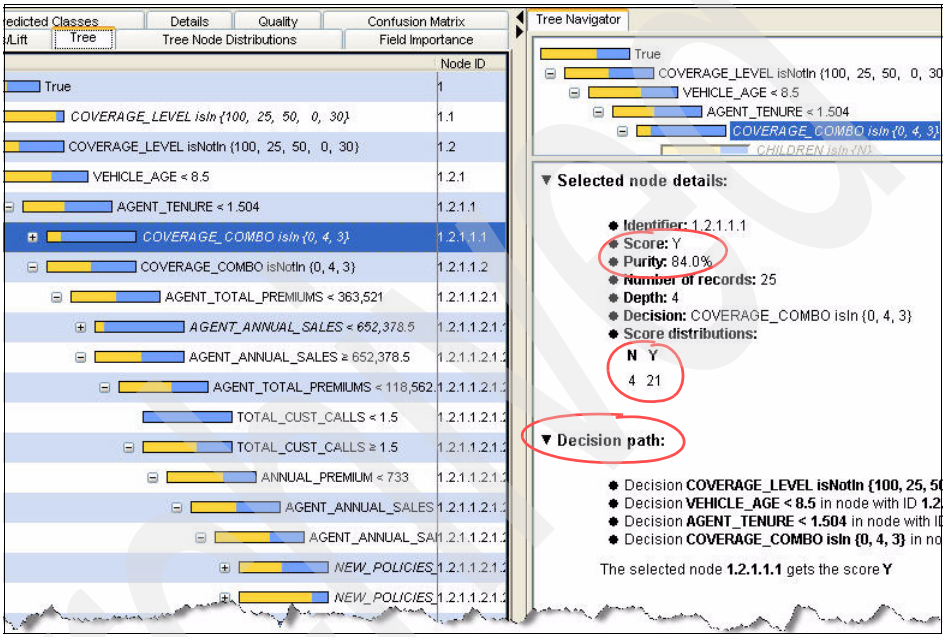


Figure 7-47 Decision tree classification example: decision tree and decision path

The relative contribution of each explanatory variable to explaining customer attrition behavior (ATTRIT) is shown in the field importance chart in Figure 7-48. In this case, the variable COVERAGE_LEVEL is the most important predictor of attrition, followed by VEHICLE_AGE, ANNUAL_PREMIUM, and so forth. The variables AGENT_AGE, TOTAL_CUST_CALLS, and NEW_POLICIES_SOLD contribute relatively little to explaining attrition, given the other explanatory variables in the model. Any variable that is highly associated with one of the variables used in the model will automatically be excluded from the model and thus will not appear in this chart.

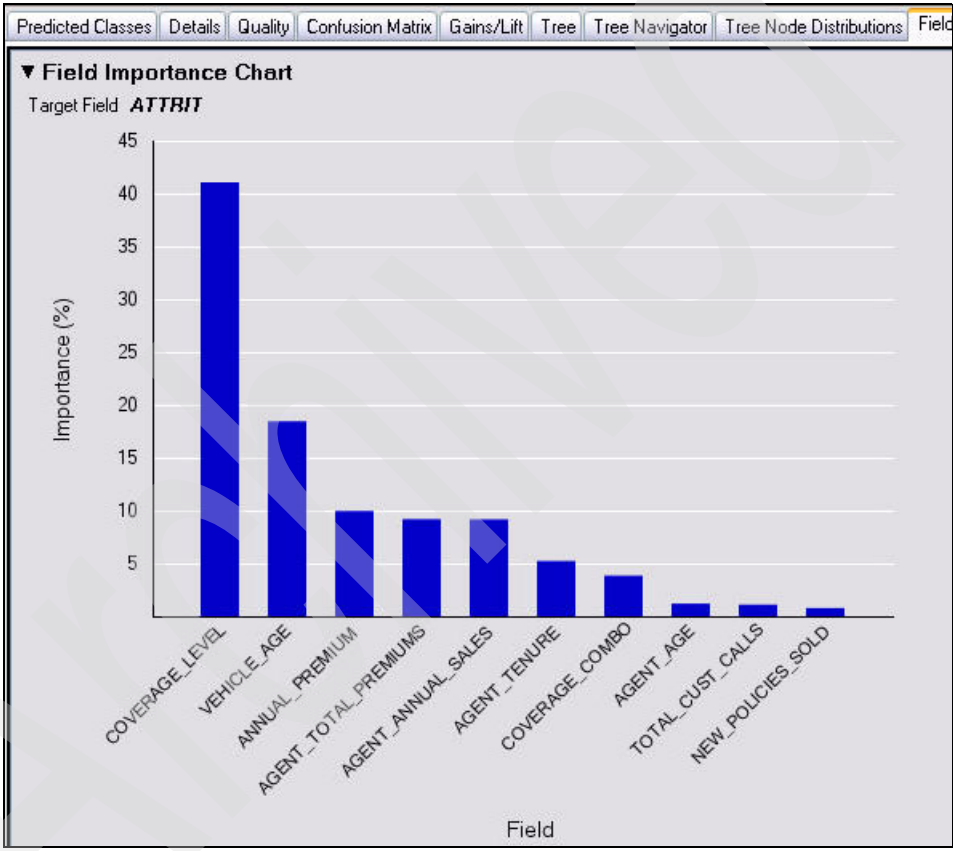


Figure 7-48 Decision tree classification example: field importance

The gains curve is a graphical representation of the model's performance. The gains curve based on the testing data is represented by the solid curve in Figure 7-49. The greater the degree of curvature of the gains curve relative to the straight dashed line below the gains curve, the better the model's performance in correctly identifying and ranking attriters. (The light dashed line above the gains curve represents the upper bound theoretically achievable by an *optimal* model having perfect accuracy.) For example, the top-ranked 30 percent of customers (horizontal axis) contains about 52 percent of the attriters (vertical axis). The ratio of the area under the model curve to the area under the optimal model curve provides a measure of the model's ranking ability.

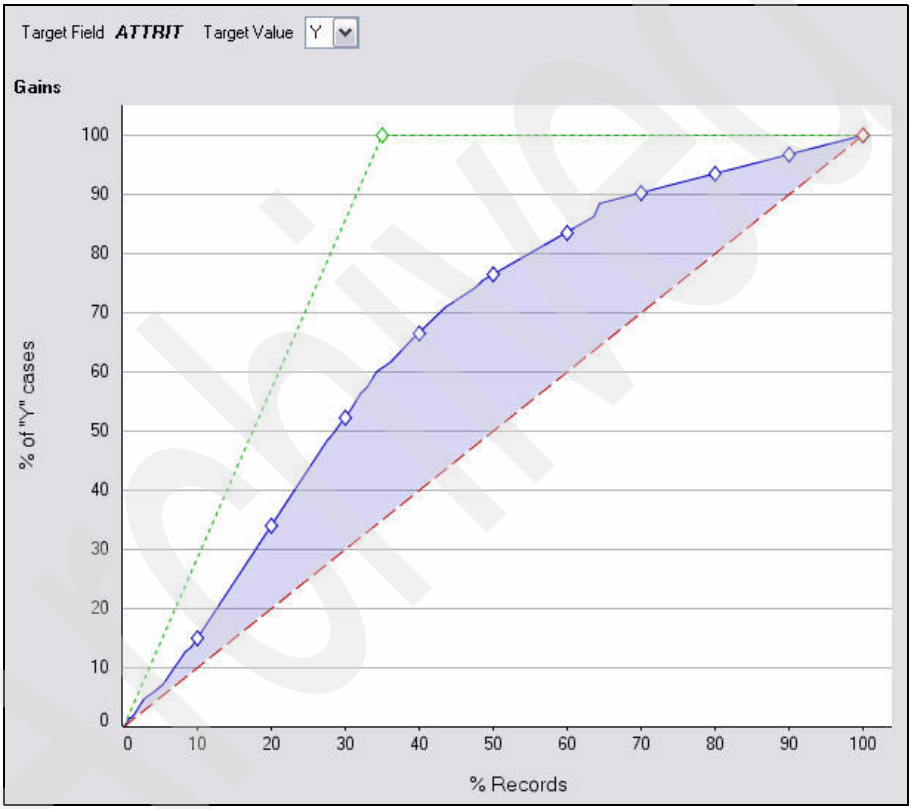


Figure 7-49 Decision tree classification example - gains curve

In Figure 7-50, the decision tree model is compared to a logistic regression model by plotting the two gains curves on the same graph. The gains curve for the logistic regression model lies slightly above the gains curve for the decision tree model over most of the range of records, that is, closer to the optimal or “perfect model” curve represented by the dashed lines at the top of the chart. This indicates that the logistic regression model performs slightly better (provides more lift) than the decision tree model in this case, especially in the top-ranked 30 percent of the customers.

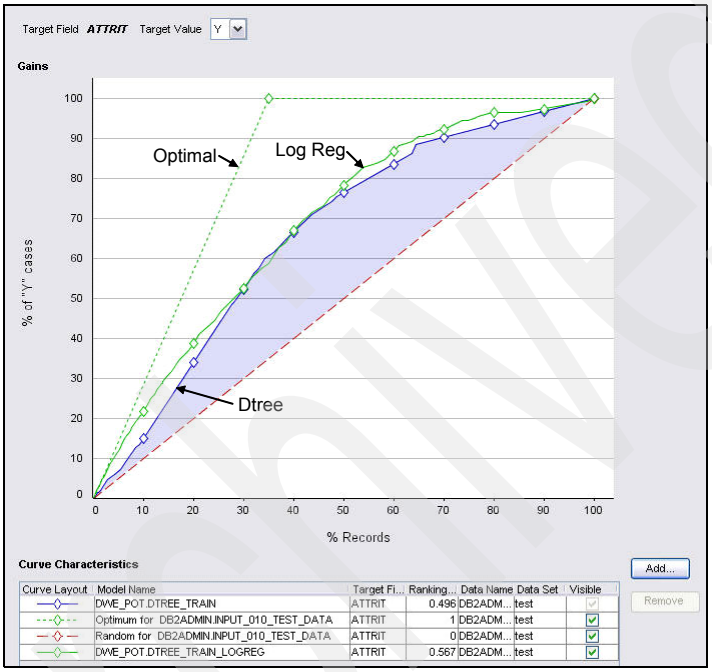


Figure 7-50 Classification example: comparing models using gains curves

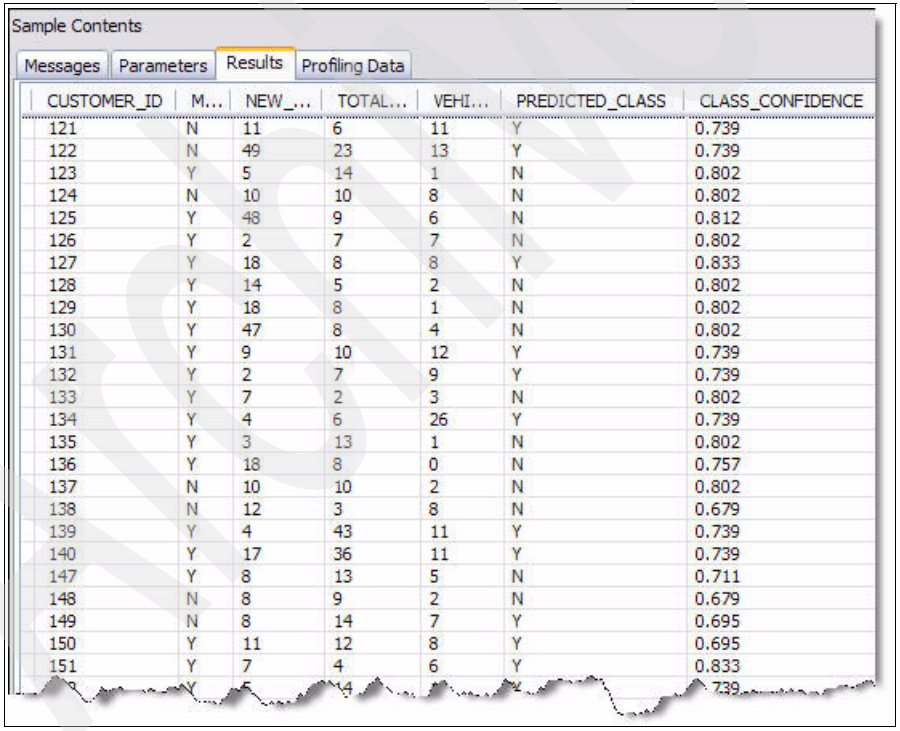
7.5.3 Scoring with a classification model

Classification models can be applied in a scoring process to answer business questions such as:

- Attrition: Will a customer voluntarily close their account?
- Disease management: Will a prospective treatment help, harm, or have no effect on a patient with a particular disease? Given a medical history and symptoms, will a patient suffer an episode of a particular disease?
- Warranty claims and failure mitigation: Will a system fail? What kind of failure will occur?

Scoring results in the output table (Figure 7-42 on page 187) include the predicted class value (Y or N) and the confidence of that prediction for each input record. The confidence value ranges from 0 (low) to 1 (high) and indicates the probability that the class is predicted correctly. For a decision tree, the confidence value for a record is the *purity* of the node containing the record. For example, if 80% of the records classified into a particular node have the value Y, then any record classified into that node has a predicted value of Y with a confidence of 0.8. An example of scored records (customers) is shown in Figure 7-51. For example, in the first row, Customer 121 is predicted to attrit (predicted class value of Y) with confidence of 73.9 percent.

Note: Other customers, for example, 122, 131, and 132, have the same predicted value and confidence, implying that they have been classified into the same node (assuming that no other node happens to have exactly the same predicted value and confidence).


Sample Contents

Messages | Parameters | Results | Profiling Data

CUSTOMER_ID	M...	NEW_...	TOTAL...	VEHI...	PREDICTED_CLASS	CLASS_CONFIDENCE
121	N	11	6	11	Y	0.739
122	N	49	23	13	Y	0.739
123	Y	5	14	1	N	0.802
124	N	10	10	8	N	0.802
125	Y	48	9	6	N	0.812
126	Y	2	7	7	N	0.802
127	Y	18	8	8	Y	0.833
128	Y	14	5	2	N	0.802
129	Y	18	8	1	N	0.802
130	Y	47	8	4	N	0.802
131	Y	9	10	12	Y	0.739
132	Y	2	7	9	Y	0.739
133	Y	7	2	3	N	0.802
134	Y	4	6	26	Y	0.739
135	Y	3	13	1	N	0.802
136	Y	18	8	0	N	0.757
137	N	10	10	2	N	0.802
138	N	12	3	8	N	0.679
139	Y	4	43	11	Y	0.739
140	Y	17	36	11	Y	0.739
147	Y	8	13	5	N	0.711
148	N	8	9	2	N	0.679
149	N	8	14	7	Y	0.695
150	Y	11	12	8	Y	0.695
151	Y	7	4	6	Y	0.833

Figure 7-51 Decision tree classification: scoring example

In Figure 7-42 on page 187, a Tree Rules Extractor operator (11) extracts the decision tree rules and writes them out to a table defined in a Table Target operator (12). This extractor is meaningful only for the decision tree classifier, not logistic regression or Naive Bayes classification. An example of the extracted rules is shown in Figure 7-52. Each row in this table represents a node in the tree. ISLEAF indicates whether the node is a terminal node called a *leaf* (1) or a node that branches further (0). CLASS is the predicted class value (Y or N in this case). SIZE is the number of records in the node. DEPTH is the depth (number of branching splits plus one) of the tree at that node. CONDITION is the decision path or rule leading to that node. For example, the last row in this table is Node 1.2.2 with predicted class value of Y and confidence of 73.9 percent, which is the node containing Customer 121 and his companions from Figure 7-51 on page 198.

Sample Contents

Messages | Parameters | Results | Profiling Data

NODEID	ISLEAF	CLASS	CONFIDENCE	SIZE	DEPTH	CONDITION
1.2.1.2.2.1.1.1.1	1	Y	0.615	13	9	VEHICLE_AGE < 8.
1.2.1.2.2.1.1.1.2	1	N	0.802	273	9	VEHICLE_AGE < 8.
1.2.1.2.2.1.1.2	1	Y	0.727	11	8	VEHICLE_AGE < 8.
1.2.1.2.2.1.2	0	N	0.595	131	7	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.1	0	Y	0.561	57	8	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.1.1	0	Y	0.447	38	9	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.1.1.1	1	N	0.679	28	10	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.1.1.2	1	Y	0.8	10	10	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.1.2	1	Y	0.789	19	9	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.2	0	N	0.716	74	8	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.2.1	1	N	0.757	70	9	VEHICLE_AGE < 8.
1.2.1.2.2.1.2.2.2	1	Y	1	4	9	VEHICLE_AGE < 8.
1.2.1.2.2.2	0	Y	0.578	64	6	VEHICLE_AGE < 8.
1.2.1.2.2.2.1	1	Y	0.938	16	7	VEHICLE_AGE < 8.
1.2.1.2.2.2.2	0	Y	0.458	48	7	VEHICLE_AGE < 8.
1.2.1.2.2.2.2.1	0	N	0.727	33	8	VEHICLE_AGE < 8.
1.2.1.2.2.2.2.1.1	1	Y	1	3	9	VEHICLE_AGE < 8.
1.2.1.2.2.2.2.1.2	1	N	0.8	30	9	VEHICLE_AGE < 8.
1.2.1.2.2.2.2.2	1	Y	0.867	15	8	VEHICLE_AGE < 8.
1.2.2	1	Y	0.739	307	3	VEHICLE_AGE >= 8

Figure 7-52 Decision tree classification: extracted rules

7.6 Regression

The following example illustrates the regression method. As in 7.5, “Classification” on page 187, the business problem is to reduce auto insurance customer attrition by identifying those customers at high risk of attriting and targeting them with retention incentives. Here, we want to predict the probability that a customer will voluntarily close their auto insurance account (attrit).

In this case, we can predict the probability of attrition by using a dependent variable (response) that is numeric with possible values of 1 (yes) and 0 (no). The predicted value, then, is the probability of “yes” for each customer.

7.6.1 Building a regression mining flow

The mining flow is shown in Figure 7-53 on page 201. A Table Source operator (1) and Select List operator (2) define the source data, which consists of one record per customer with columns representing customer attributes, attrition response, and agent characteristics, just as in the classification example in 7.5.3, “Scoring with a classification model” on page 197. A Random Split operator (3) divides the input data into training and testing sets. The training data flows from (3) to the Predictor operator (4), which executes the appropriate predictive mining algorithm (regression in this case, since the dependent variable is numeric) and sends the trained model to a Visualizer operator (5). The testing data from (3) and the model from (4) flow to the Tester operator (6), which applies the model to the testing data and passes the test results to another Visualizer operator (7). The model from (4) and a Table Source operator (8) containing new records are passed to a Scorer operator (9), which applies the model to the input records to generate the scoring information and then write the results to an output table specified in a Table Target operator (10). Available scoring results include the numeric value predicted for the record (in this case, the predicted probability that a customer will attrit), the confidence that the predicted value lies within a user-specified interval, and the standard deviation around the value predicted for each record (which indicates the expected range of the actual value).

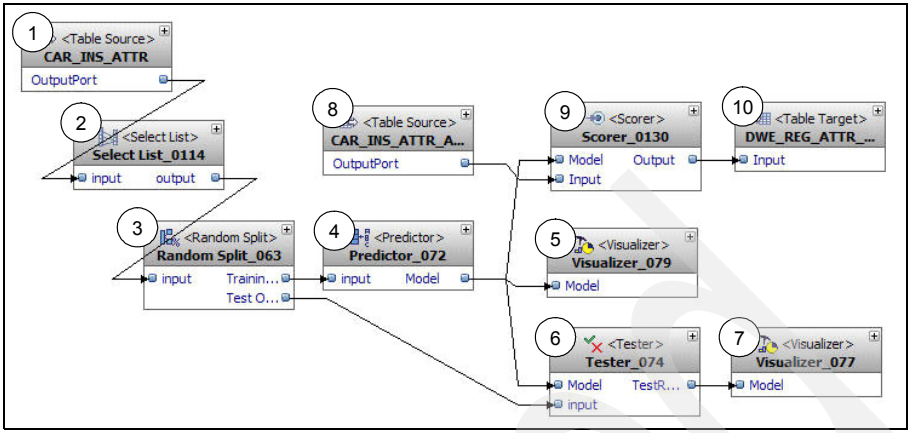


Figure 7-53 Mining flow for regression

- The Predictor operator (4) offers the choice of three regression algorithms: Transform Regression, Linear Regression, and Polynomial Regression. Transform Regression (the default method) is an accurate, fast algorithm developed by IBM Research. It iteratively combines stepwise Linear Regression and nonlinear field transformations and generally yields better models than linear or Polynomial Regression. To avoid overfitting the model to the training data, it automatically uses part of the training data for validation during model creation. Linear Regression assumes a linear relationship between the explanatory fields and the target field (dependent variable) and produces models that represent intrinsically linear equations. It can use either stepwise regression or maximization of adjusted R^2 to produce the optimal model. Polynomial Regression assumes a polynomial relationship and uses Linear Regression to produce intrinsically linear models containing polynomial terms. Subsets of explanatory variables are selected using the adjusted R^2 to find the optimal model. If the user sets the maximum degree of polynomial to 1, then the Polynomial Regression algorithm and the Linear Regression algorithm produce identical models. If the maximum degree is set to its maximum value of 3, the Polynomial Regression algorithm tends to overfit the training data and may not fit testing data or new data well. For more information about these regression techniques, see *Transform Regression and the Kolmogorov* Superposition Theorem, by Pednault, found at `http://www.siam.org/meetings/sdm06/proceedings.htm` and *Elements of Econometrics*, *2nd Ed.*, by Kmenta.

7.6.2 Interpreting a regression model

The regression visualizer presents information to evaluate a regression model. In this example of predicting the probability that a customer will attrit, the dependent variable is ATTRIT_01 with possible values of 1 (yes) and 0 (no). Given the business question, we are particularly interested in predicted values close to 1.

The regression visualizer presents information to evaluate a regression model and its testing results. Similar to the classification visualizer, the regression visualizer reports the model quality, field importance, and a gains curve. Model quality for the testing set is shown in Figure 7-54. Interpretation of the metrics is the same as for the classification in 7.5.2, “Interpreting a classification model” on page 190.

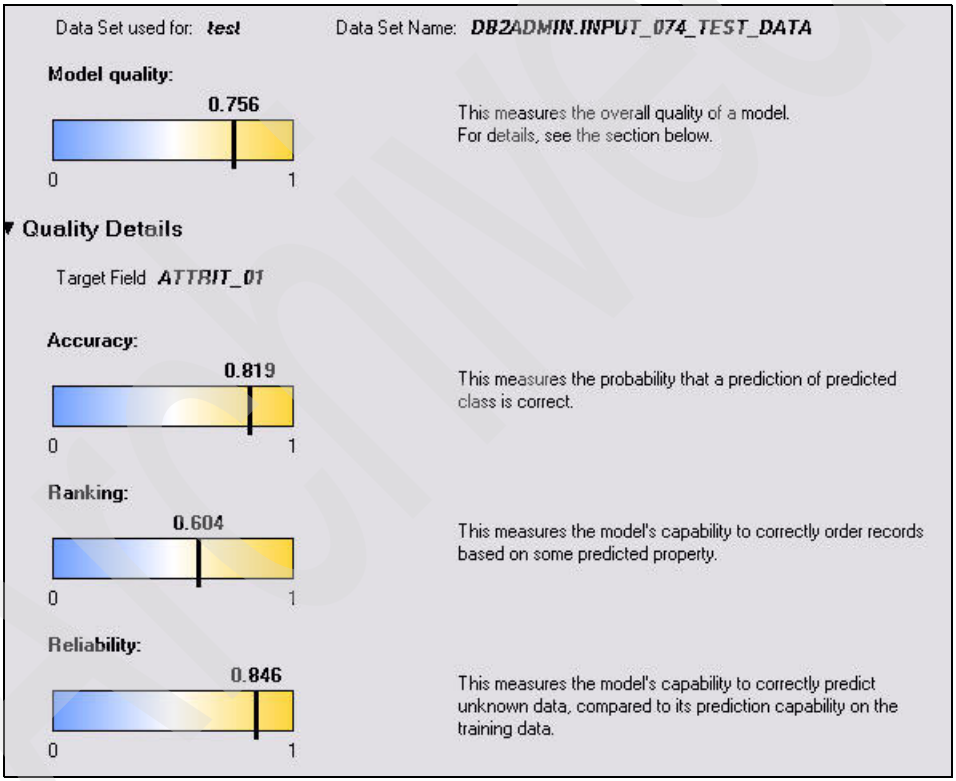


Figure 7-54 Regression example: model quality

The field importance chart from the training results visualization is shown in Figure 7-55. In this case, AGENT_TENURE is the most important predictor of the likelihood of attrition, followed by AGENT_ANNUAL_SALES, AGENT_TOTAL_PREMIUMS, and so on. (Clearly, this model indicates that the agent's experience and productivity are very important to the customer's decision regarding attrition.) As with classification, any variable that is highly correlated with one of the variables used in the model will automatically be excluded from the model and thus will not appear in this chart.

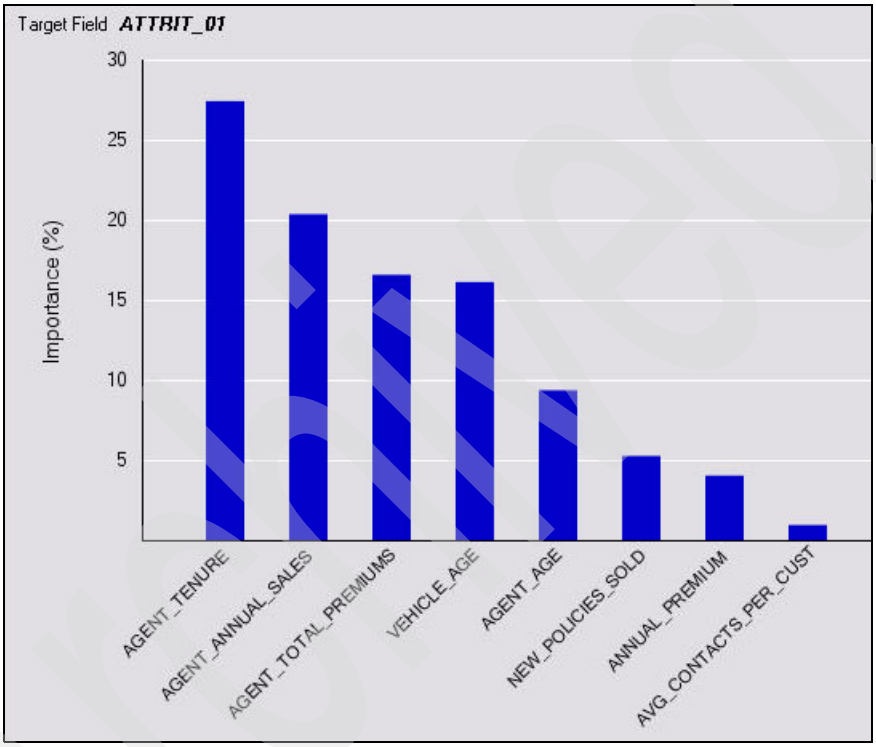


Figure 7-55 Regression example: field importance

The gains curve based on the testing data is represented by the solid curve in Figure 7-56. As with classification, a greater degree of curvature of the gains curve indicates better model performance in correctly identifying and ranking attriters. For example, the top-ranked 20 percent of customers (horizontal axis) contains 40 percent of the attriters (vertical axis).

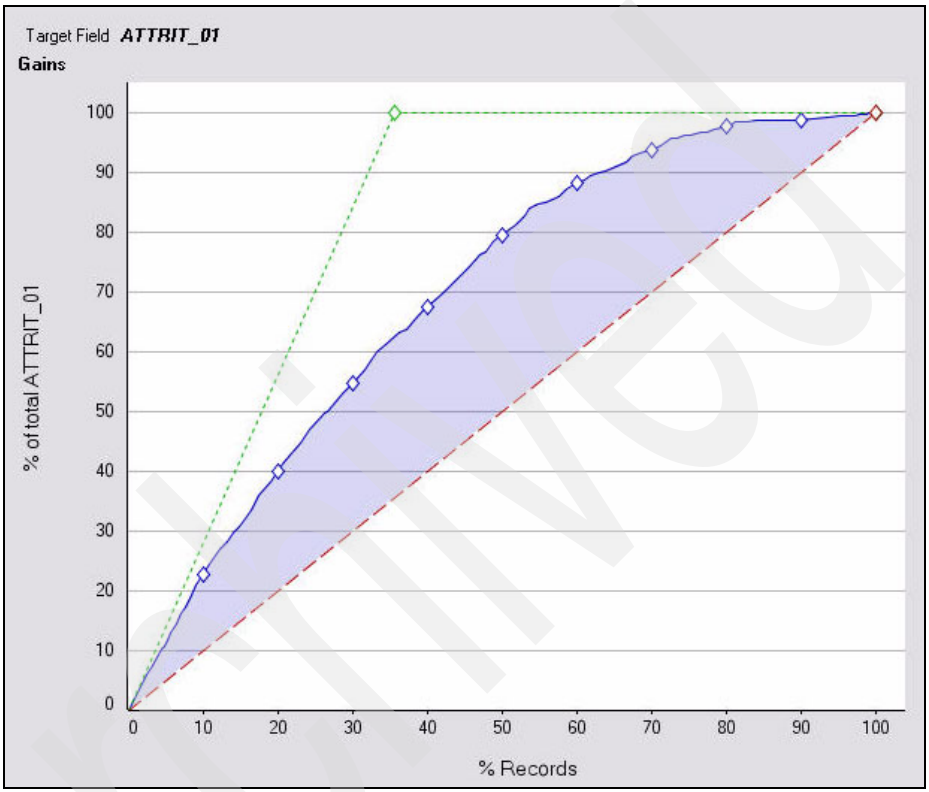


Figure 7-56 Regression example: gains curve

7.6.3 Scoring with a regression model

Regression models can be applied in a scoring process to answer business questions such as:

- Attrition: What is the probability that a customer will voluntarily close his account?
- Customer relationship management: What is the expected lifetime value of a customer? What is the probability that a customer will respond to a particular offer?

- Healthcare: What is the likelihood that a person will develop a particular disease? What is the expected rehabilitation time for a particular hip-replacement patient?

Scoring results for this customer attrition example are shown in Figure 7-57. As suggested by the relative field importance chart (Figure 7-55 on page 203), agent tenure and vehicle age are two of the most important predictors of a customer's propensity to attrit. For example, in the first row in the table, Customer 2 has an old vehicle (16 years) and an inexperienced agent (tenure of 1 year), and his predicted propensity to attrit is 1. Customer 28, on the other hand, has a new vehicle (0 years) and a more experienced agent (tenure of 4 years), and he has a predicted propensity of 0 (where the predicted value of -0.06 is interpreted as 0).

AGENT_ANNUAL_SALES	AGENT_TENURE	AGENT_TOTAL_PREMIUMS	CUSTOMER_ID	VEHICLE_AGE	PREDICTED_VALUE
641145	3	22769	1	6	0.309
481151	1	70061	2	15	1
1284112	2	201409	3	4	0.822
1011243	1	114519	4	7	0.619
1236251	3	63628	5	9	0.474
551010	1	55764	6	11	0.782
1432695	1	65431	7	8	0.551
802667	1	107077	8	3	0.54
1039628	2	87431	9	10	0.695
1355599	1	224141	10	6	0.675
1055301	1	74511	11	11	0.76
1922000	1	147537	12	18	0.921
1715409	1	128854	13	10	0.723
880894	1	160525	14	25	1
1373644	1	144245	15	12	1
1346080	2	65974	16	0	0.17
1445586	1	69574	17	10	0.88
2255450	2	141724	18	3	0.306
1242665	2	119990	24	0	0.449
1039966	1	91655	25	11	0.792
1440726	1	93956	26	5	0.374
1083911	1	44713	27	9	0.517
1985848	4	85745	28	0	-0.06
1516246	1	63780	29	25	0.277
1894117	2	286417	30	2	0.597
1115734	4	47841	31	1	-0.019
1600487	1	148187	32	10	0.781
1301528	4	68932	33	0	-0.017
796989	4	72912	34	0	0.086
1561079	1	189343	35	9	1
1162236	1	184404	36	3	0.674
2255450	2	141724	37	7	0.495

Figure 7-57 Regression: scoring example

7.7 Oversampling to create an enriched training set

Predictive models are intended to make predictions about some response or outcome, but sometimes the response value of primary interest (for example, “Yes” for a response variable with possible values of “Yes” and “No”) occurs rarely in the population. For example, a weather model to predict rainfall in Texas might be 99 percent accurate simply by always predicting “no rain,” but such a trivial model tells us nothing about what we really want to know, which is when it will rain. By drawing a sample from the population whereby we oversample the response value of interest relative to its true occurrence in the population, we can create an enriched sample for training a predictive model. By using the enriched training set to build the model, we may be able to do a better job of modeling the outcome of interest. The model is then validated using a testing set with the correct population proportions of the response values.

Although the “stratified sampling” option can be used in a Random Split operator to generate a training set with approximately equal proportions of the values of the stratification variable (which could be the response variable if it is categorical), as discussed in 7.5.1, “Building a classification mining flow” on page 187, an enriched sample with any desired proportion of the response value of interest can be created using a mining flow. An example of oversampling the “Yes” outcome (indicating a subsequent episode) from a set of congestive heart failure (CHF) patients to produce a training set containing approximately half “Yes” and half “No” outcomes is shown in Figure 7-58. This procedure assumes that the source table contains one row per individual (as for clustering, classification, or regression) rather than multiple rows per individual (as for associations or sequences).

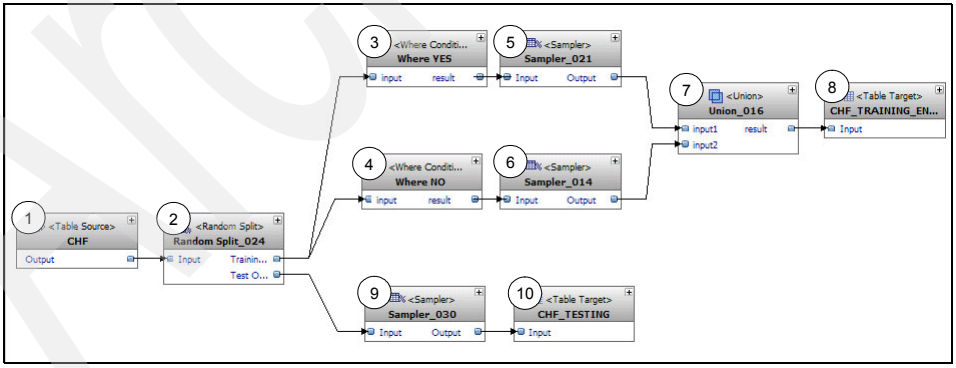


Figure 7-58 Mining flow for oversampling to create an enriched sample for training

In Figure 7-58 on page 206, the source data (1) contains patient attributes and the dependent variable GT90DAYS having possible outcomes of Y (yes) and N (no). The source data is divided into two groups of approximately the same size using a Random Split operator (2). The first group is then divided into two groups containing all of the Y records and all of the N records, respectively, by the Where Condition operators (3) and (4), as shown in Figure 7-59 and Figure 7-60 on page 208.

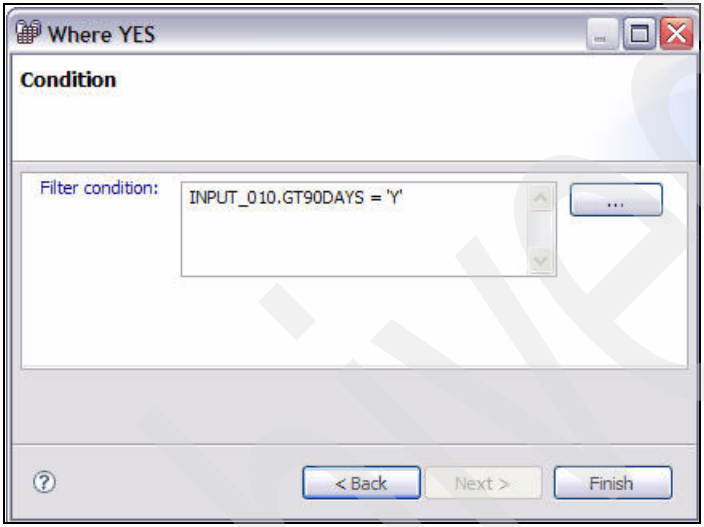


Figure 7-59 Where condition for patients who have another CHF episode

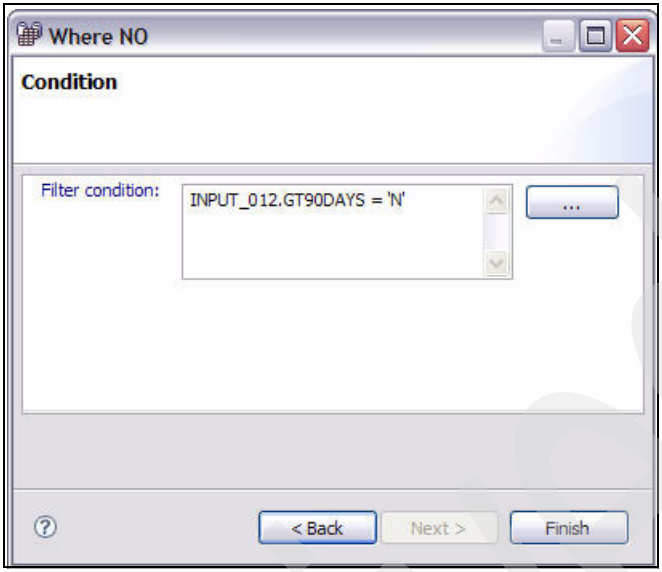


Figure 7-60 Where condition for patients who do not have another CHF episode

Sampler operators (5) and (6) select random samples of different sizes from (3) and (4), respectively, as shown in Figure 7-61 on page 209 and Figure 7-62 on page 209. In this case, the Sampler operator (5) takes all of the Y records from (3), while (6) takes less than 100 percent of the N records from (4) such that the counts of Y and N are approximately the same, given the proportions of Y and N in the source data (1). The two samples from (5) and (6) are then combined by a Union operator (7), and the resulting enriched training set containing approximately equal numbers of Y and N records is written to an output table (8). By changing the sampling rate in (5) or (6), the relative proportions of Y and N in the enriched training set can be controlled. The second group from (2) is passed to a Sampler operator (9), which in this case takes all of the records in this group, and the resulting testing set is written to an output table (10). The testing set contains Y and N records in the same proportions as the source data (1).

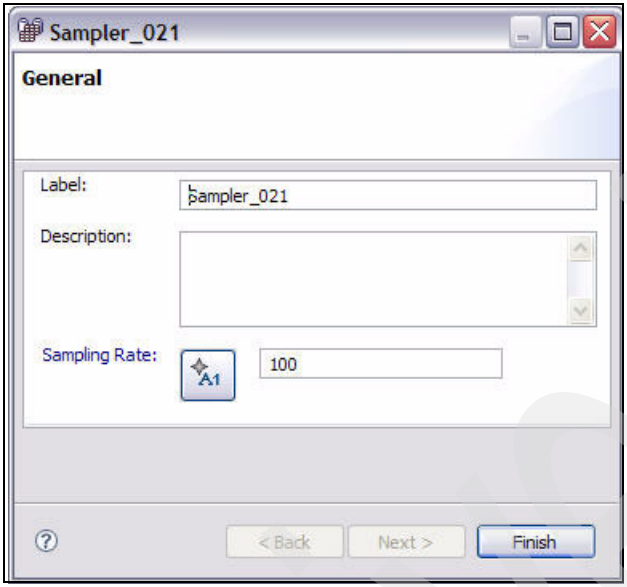


Figure 7-61 Sampling 100 percent of the GT90DAYS = 'Y' records

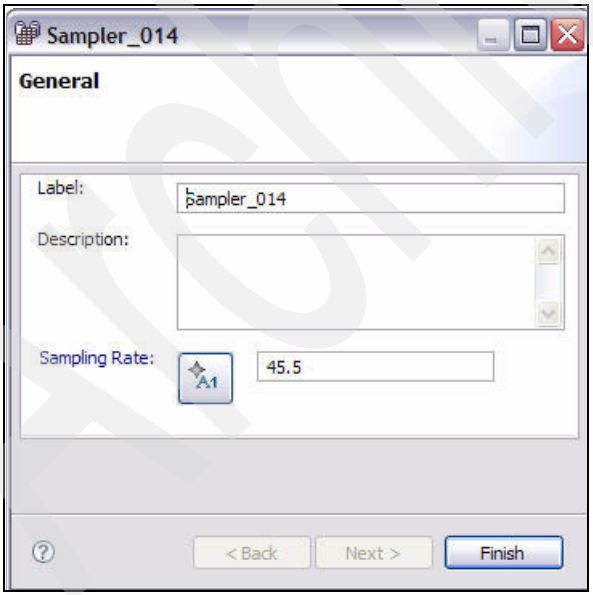


Figure 7-62 Sampling 45.5 percent of the GT90DAYS = 'Y' records

The operations shown in Figure 7-58 on page 206 are illustrated further by the example in Figure 7-63. This example shows record counts and proportions for the source table (1), the enriched training set (8), and the testing set (10). Note that the training set has approximately equal proportions of Y and N records, while the testing set has approximately the same proportions of Y and N records as the source data. Because the training and testing sets include all of the Y records, the total count of Y in the training and testing sets equals the number of Y in the source data. Because the training set was enriched by excluding some of the N records, however, the total count of N in the training and testing sets is less than the number of N in the source data.

	Source Table (1)		Enriched Training (8)		Testing (10)		Training + Testing
	Count	Percent	Count	Percent	Count	Percent	Count
N	2,011	69%	466	49%	1,004	70%	1,470
Y	916	31%	478	51%	438	30%	916
Total	2,927	100%	944	100%	1,442	100%	2,386

Figure 7-63 Oversampling example

7.8 Troubleshooting tips

Although troubleshooting runtime problems with mining flows is usually straightforward, some problems can be difficult to identify and resolve when they are encountered for the first time. In this section, we address two such problems that may arise when executing mining flows.

7.8.1 Cleanup after abnormal termination of a mining flow

Abnormal termination of a mining flow during execution may leave the system in a state that generates an error message that the program is still running, which prevents the flow from being re-executed. A cleanup process cancels the execution and removes objects created by the mining flow (for example, views, models, or test results), after which the mining flow can be re-executed. This process consists of calling three procedures, each of which uses a model name as a parameter.

Perhaps the simplest way to run the cleanup process is to create a script using a text editor and then run the script on a DB2 command line. The script consists of the following three lines, where the schema is IDMMX and the model name is My Mining Model:

```
call IDMMX.SetTaskStopped ('My Mining Model');
call IDMMX.CancelTask ('My Mining Model');
call IDMMX.CleanUpTask ('My Mining Model');
```

To execute this script, open a DB2 command window and enter the following commands to connect to the database called MYDATABASE and run the script called cleanup.sql:

```
db2 connect to MYDATABASE
db2 -tvf cleanup.sql
```

7.8.2 Clearing a full DB2 transaction log

When writing results to an output table in a mining flow, a typical practice is to check the **Delete previous content** box in the Table Target operator's properties. This selection is equivalent to the DB2 command:

```
delete from MYSCHEMA.OUTPUT_TABLE
```

This command deletes the rows from the output table MYSCHEMA.OUTPUT_TABLE without dropping the table itself. As each row is deleted, it is written to the DB2 log. Although this normally presents no problem, sometimes the number of output rows can be exceedingly large, such as when a cross-join of large tables yields a cartesian product. In such a case, the log may fill up before execution has completed, resulting in a runtime error (transaction log full). Increasing the number or size of the DB2 log files may not solve the problem.

Once the culprit operation, for example, a cross-join, has been identified and fixed, the problem is then to delete the rows from the DB2 log. A simple process to do this is as follows:

- Using a text editor, create a file called, say, XXX.TXT, which contains no rows.
- From a DB2 command window, run:

  ```
  db2 connect to MYDATABASE
  db2 load from XXX.TXT of del replace into MYSCHEMA.OUTPUT_TABLE
  ```

This process replaces all of the rows in the output table with nothing. So when the mining flow is re-executed, the DB2 log is cleared and the flow will execute.

8

Deploying a data mining solution

So far, the chapters of this book have taken the journey of data mining from initial demystifying concepts through how to create a data mining model using the DB2 Warehouse 9 Design Studio tooling. In this chapter, we introduce a few techniques with which data mining solutions can be deployed throughout the enterprise. Deployment may be as simple as automating the data mining process to update models when the data changes or as complex as implementing embedded, high-speed scoring to make real-time cross-sell recommendations during a customer checkout at a register.

Many data mining projects stop at the point of showing the data mining results in a presentation to an executive, for making a one-time strategic decision. The real return, however, comes when a company is able to leverage data mining in the day-to-day running of the business, by deeply integrating data mining into the business processes. In other words, creating a data mining solution rather than just executing data mining runs. This moves data mining from the back office out to line-of-business users through their desktops or Web-based portals so they can make decisions and act much faster.

8.1 Embedding data mining into the business process

Businesses today are looking for ways to make data mining accessible to large numbers of line-of-business analysts and decision makers throughout the enterprise. *Embedded data mining* is the concept of delivering data mining to business analysts through portals, reporting tools, and customized applications tailored to specific business areas, all within the database environment. Embedded data mining components implemented as database extenders for model building and application facilitates the development and enterprise-wide deployment of data mining-based solutions by enabling data mining functions to be integrated into BI tools. Because data mining capabilities are implemented as database extenders, any reporting or query tool that can use SQL can become a data mining platform. Business analysts who are not accomplished statisticians can readily bring the power of data mining to bear on many different business problems.

For embedded data mining, the data warehouse is the single, centralized, and comprehensive data source for integrated BI analytical and related functionality. Through centralized management of data resources, many of the problems, errors, and costs inherent in moving and duplicating data can be avoided. Working directly in the database also enables the use of high-throughput, scalable data warehouse platforms for BI analytics.

BI analytics can be effectively supported by domain- or project-specific data structures within the overall data warehouse, reducing data manipulation costs. For example, to support an ongoing customer segmentation project, a data mart could be established to regularly perform the required data transformations to populate and refresh a table containing the customer data in the correct structure for the analysis. Another data mart could likewise support market basket analysis, which requires a different data structure than customer segmentation. By maintaining and refreshing the data for different analytical uses, data marts can greatly reduce the overall analysis time.

Bringing data mining capabilities directly into the relational database used for the data warehouse also helps the IT staff as data remains in the data warehouse. Database administrators do not have to worry about the implications of having data stored in files outside of the data warehouse environment, such as data backup, data protection and data auditing. Using existing management tools and techniques developed for the data warehouse helps to minimize the overhead costs of managing and controlling data mining applications in a production environment.

The need to integrate mining into the database, with both embedded data mining and traditional data mining (working outside the central database environment), has led to the creation of standards establishing the Predictive Model Markup Language (PMML) and a standard SQL interface to data mining capabilities. PMML provides a way to express a data mining model as an XML object that can be ported between analytical environments without having to rebuild or re-code the model. This capability enables analysts to create data mining models and deploy them in their native forms in the data warehouse where they can be used by applications. PMML also facilitates model refreshing without recoding and retesting, greatly reducing the time to implement refreshed models and ensuring that business analysts are able to use up-to-date models in their decision making. For more information about PMML, refer to the Data Mining Group and to the IBM Redbooks publication *Leveraging DB2 Data Warehouse Edition for* Business Intelligence, SG24-7274.

8.2 Data mining application types

The opportunity to embed data mining functions in an enterprise is vast and largely unexplored by most companies today as embedded data mining is a relatively new concept. Nonetheless, the desire to push decision-making out to the line-of-business is very high indeed. This interest in easier ways to bring data mining to large numbers of users is growing at a tremendous rate. An additional benefit of embedding data mining into applications for companies that have a data mining staff is that the data mining professionals can be relieved of doing many common, production-oriented data mining tasks and can concentrate on solving tougher business problems.

We classify data mining applications into two categories:

1. *On demand data mining*: Applications that allow the users to create, refresh, or visualize data mining models.
2. *Real-time scoring*: Applications that allow the users to apply a data model to new data.

What makes these type of data applications much easier to create is the fact that data mining in embedded into the DB2 database engine and is accessible using the SQL language. See 8.3, “Techniques for embedding data mining solutions” on page 227 for more information about the DB2 Warehouse 9 data mining application programming interfaces.

8.2.1 On demand data mining

In this category of data mining, applications are developed to allow data mining models to be created or refreshed without the intervention of a data mining professional. These applications could be batch-oriented processes that are scheduled as part of the data warehouse update cycle or may be sophisticated user applications invoked from a Web portal. No matter how these applications get started, they will use the data mining SQL functions to perform one or more tasks to accomplish the business objective and to integrate data mining with normal application functionality.

Back-room applications

These type of applications would generally be batch-oriented and would likely be managed as part of the IT operations. They may be invoked automatically based on a schedule or manually executed by an operator.

A typical use for a back-room data mining application is to refresh a data mining model based on some event or on a regularly scheduled basis. For example, if a data warehouse is updated nightly, a refresh of data mining models would be integrated into the ETL process. Or perhaps a data mining model in an OLTP environment is automatically refreshed periodically throughout the day.

The most common use in a back-room application is to score new data as it flows into the data warehouse or to do a large-scale scoring (or re-scoring) of data in the data warehouse. This could be an initial scoring of the data, based on an updated data mining model, or a significant change in the underlying data. For example, in a customer profiling application, new customers would be scored as their records flow into the data warehouse to assign them to the appropriate customer segment. There may also be a need to rescore all customers in the data warehouse if there has been a significant change to the underlying data either recently or a accumulation of changes over time.

DB2 Warehouse tooling

As seen in Chapter 5, “DB2 Warehouse tooling for data mining” on page 51, the DB2 Warehouse 9 Design Studio has graphical tooling for creating data mining runs to build and refresh models. These are called Mining Flows. If these Mining Flows are integrated into Control Flows, perhaps with other types of processing, they can be deployed to the DB2 Warehouse 9 Administration Console for one or more production environments where they can be executed, perhaps through a schedule, to build or refresh data mining models.

In Figure 8-1, a Design Studio mining flow has been developed to refresh a store profile mining model. This certainly can be directly executed manually from the Design Studio. However, this is a manual task. The need to refresh the model may be dependent on an update of the warehouse data. Integrating the mining flow into a control flow using a mining flow operator can sequence the model refresh within the context of the update of the data warehouse and it can be automatically executed in the correct order.

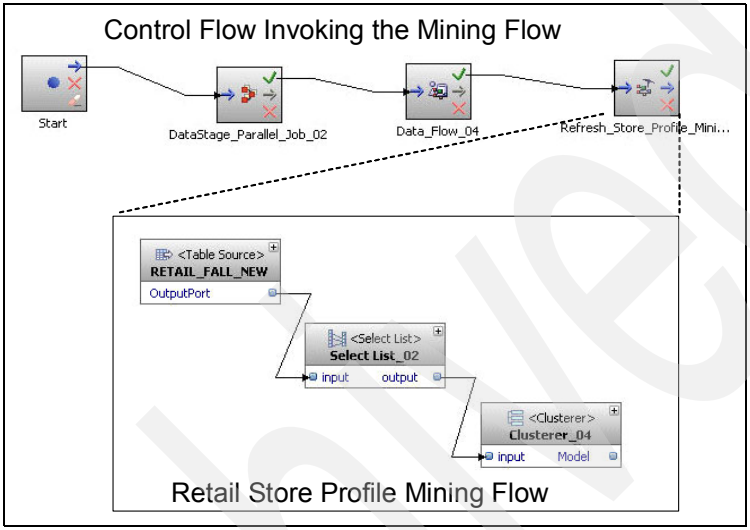


Figure 8-1 Integrating a mining model refresh into ETL processing

The most common type of backroom mining application is driven by the need to do large-scale batch scoring whether it is scoring a risk factor for new customers as they flow into the data warehouse or perhaps re-scoring the risk factor for all customers after a warehouse update. In the Design Studio, this will be accomplished by using the scoring operator found in a Data Flow, which uses an existing mining model and applies it to a set of data.

Other tooling

Using the DB2Warehouse 9 tooling is just one way to be able to create these backroom mining applications. Using one or more of the supported Application Programming Interfaces (APIs), batch-oriented, backroom mining applications can be developed using a wide variety of other tools, from simple scripts to Java applications or even embedded into ETL tools such as DataStage™. See 8.3, “Techniques for embedding data mining solutions” on page 227 for an introduction to the APIs.

Front-room applications

While doing mining in the backroom is fine, there is a huge potential return when mining capability can be pushed out of the IT shop and out of the corporate offices to line-of-business users. In this category of on demand mining, this simply means building an application by which a user would be able to execute and see the results of a data mining run.

These applications can cross the spectrum of sophistication from providing a very simple interface for a knowledge worker to call data mining runs to fully embedding data mining into a normal business application that is used by a line-of-business worker who has absolute no knowledge that they are doing data mining.

One application may simply provide a list of mining jobs to execute which, after execution, might display the visualization tool. Another application might provide the capability to allow a buyer to perform market basket analyses. Figure 8-2 shows a market basket analysis application delivered over the Web. The user simply selects the product categories and the date range and some optional tweaking parameters and clicks one button. The applications uses the SQL-based API that performs the data mining run and presents the results of the market basket analysis in a simple table format.

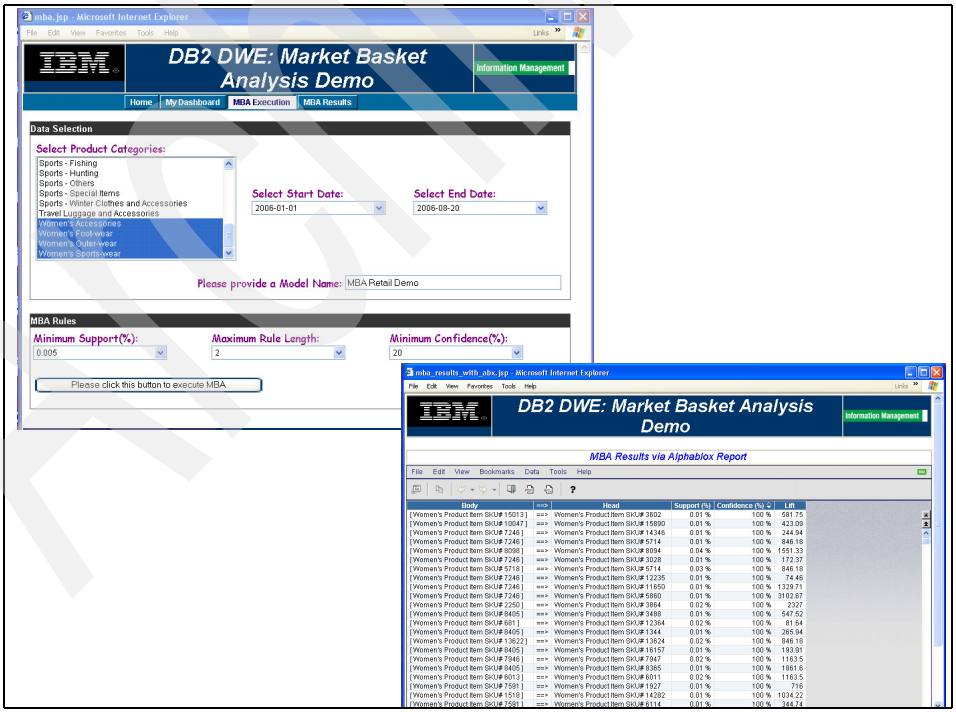


Figure 8-2 Web-based Market Basket Analysis application

8.2.2 Real-time scoring

Section 8.2.1, “On demand data mining” on page 216 discussed applications that utilized data mining functions that typically take some time to execute from many seconds to perhaps hours and may mine or score many thousands or even millions of records. There is, however, an entire class of applications that need to apply a data mining model to just one, or a few, records on the fly in real time. This is a case where having data mining functions embedded in the relational database is extremely powerful.

There is an unlimited number of application examples for doing real-time scoring across many industries.

Anyone who has done online shopping is familiar with real-time data mining. As one is browsing items or adding items to his shopping basket, many of these sites will make recommendations based on the behavior of others who looked at or purchased the same item. This is real-time scoring of a data mining model.

This cross-sale capability can be extended to brick-and-mortar stores as well by embedding data mining into the store systems that control the registers. For example, a department store is very interesting in increasing the amount that a customer spends during a visit. One way to do that is to entice the customer to visit other departments during the visit. Therefore, during checkout, this customer and his purchases can be scored against a data mining model and a personalized incentive to visit other departments in which they have a high likelihood of being interested.

Many industries that need to make an immediate decision based on some type of risk value. This may be pricing an insurance policy or approving a loan. It is extremely common today to use the Web or an automated phone system to price insurance policies or get and instant decision for a loan request. Again, this is real-time data mining in action.

Also, in the financial arena, real-time scoring may be embedded into ATM or credit-card processing systems to detect potential fraud. Many people have experienced a refusal of a credit card or ATM transaction and, when calling customer service, have found out that the card was flagged because of spending patterns outside the norm.

Real-time scoring does not apply just to consumer transactions. Real-time scoring can be used by manufacturing companies to detect quality issues in a manufacturing process. There are even studies in progress to determine how real-time data mining might be used in the Intensive Care Unit of the future to monitor streaming biomedical data to create alerts to the medical staff.

8.2.3 Tools for developing data mining applications

Earlier in this chapter, we discussed tools that can be used to create batch-oriented backroom applications to refresh models and to massively score data in the warehouse. This section introduces tools that can be used to deliver data mining applications to users for on demand mining and real-time scoring.

Alphablox application

Alphablox is a set of tools to help develop Web-based applications to deliver analytics. Alphablox components can be used to develop complete dashboards or applications but it is extremely powerful when there is a requirement to embed analytics into normal line-of-business Web applications. See Chapter 5, “DB2 Warehouse tooling for data mining” on page 51 for more information.

Utilizing the SQL data mining functions along with the powerful Alphablox components in a jsp-based Web application helps the Web application developer to be much more productive when embedding data mining into applications. Using the market basket analysis example in Figure 8-2 on page 218, Alphablox visual components are used for gathering the input and for displaying the resulting report. Non-visual Alphablox components are used to invoke the underlying data mining routines and to gather the results.

The market basket analysis application is basically a stand-alone mining application delivered through the Web, but that is not the only method of delivering data mining functions to users with Alphablox. Alphablox can also be used to develop dashboards into which data mining can be embedded, as seen in Figure 8-3.

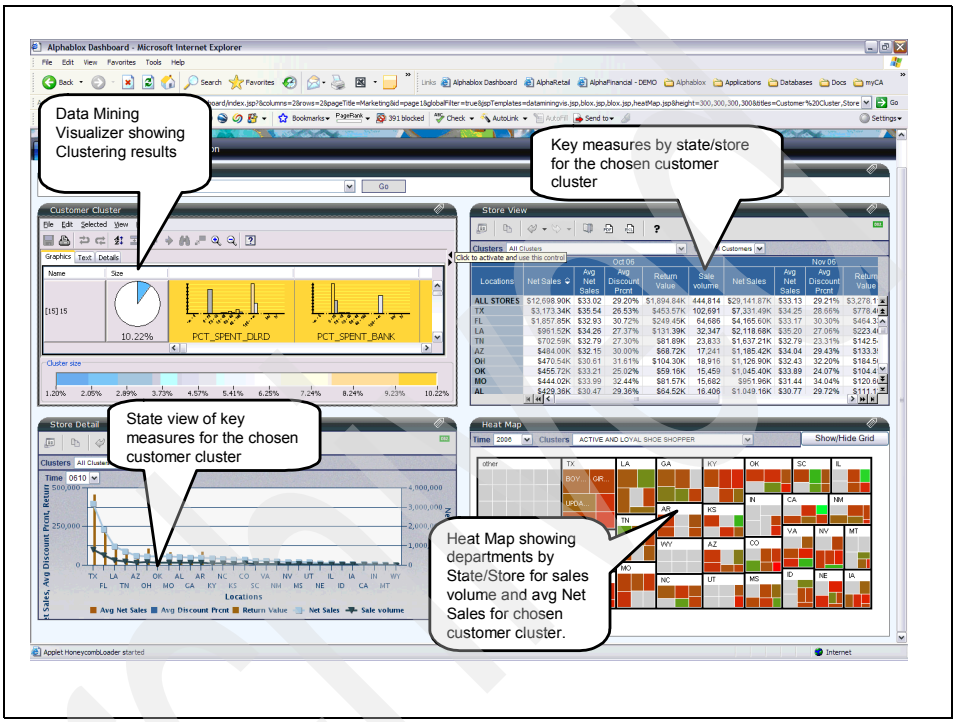


Figure 8-3 Marketing dashboard with embedded data mining

In this dashboard, the user wants to view key indicators of a cluster of similar customers. This dashboard presents the results of a data mining clustering run to create sets of customers with similar behaviors. The user utilizes the data mining visualizer to find and select a cluster of customers of interest. The application then displays various Alphablox components to see key information about that cluster of customers and the departments and stores associated with that cluster. The user does not necessarily know that they are actually utilizing data mining.

Alphablox applications can also be imbedded into Web portals and use portal-to-portal communications techniques. Utilizing Alphablox, data mining can also be embedded into a portal. For example, Figure 8-4 shows a workbench for a grocery store manager. It would probably be safe to say that most grocery store managers really do not care if they are using data mining or not. They are so busy and just want their information presented to them the way they want it at the time they need it. In this example, a store manager is alerted to an out-of-stock (OOS) condition and needs to discover why. Just using a simple point and click portal-based application, the store manager can quickly isolate the root cause of the OOS condition and take quick action. The store manager is not aware of, and has no interest in, the fact that the application integrates all types of sophisticated data analysis techniques including data mining. To the manager, the task of quickly solving the OOS condition is what matters.

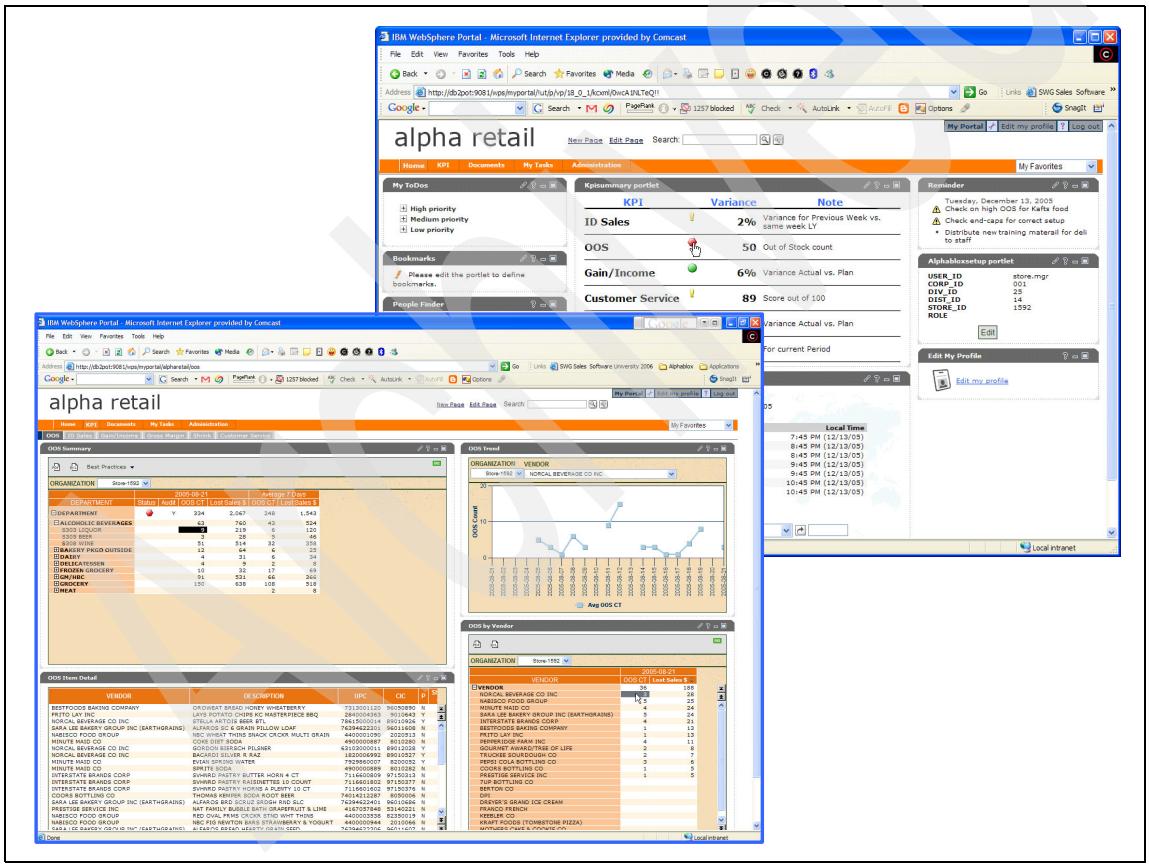


Figure 8-4 Grocery store manager portal

Miningblox application

Miningblox is a recent addition to the DB2 Warehouse tools. Miningblox is an extension to Alphablox to provide data-mining specific functionality in the form of new Alphablox tags. At the core of a Miningblox application are mining flows that are developed within the DB2 Warehouse Design Studio and deployed to a production environment through the DB2 Warehouse Administration console. Miningblox tags are then used to select input data, invoke the mining flow to process the data, and then present the results to the user. There are also functions to help manage and administer the mining runs.

Using a combination of standard Alphablox and Miningblox tags allows fully customized data mining applications to be developed much quicker. In addition, development time can be further reduced by using the Miningblox Application Wizard in the DB2 Warehouse Design Studio. The wizard assists with the generation of a data mining application based on templates.

Whether a Miningblox application is created using the Miningblox Application Wizard or is custom-developed, the structure of the application is still the same as shown in Figure 8-5. It consists of the following steps:

1. The user selects input data and executes the mining flow.
2. The mining flow executes asynchronously. In the meantime, the status can be monitored using the task management tags. The list of completed and executing task can be displayed for the user.
3. Once the mining flow has executed, the results can be displayed.

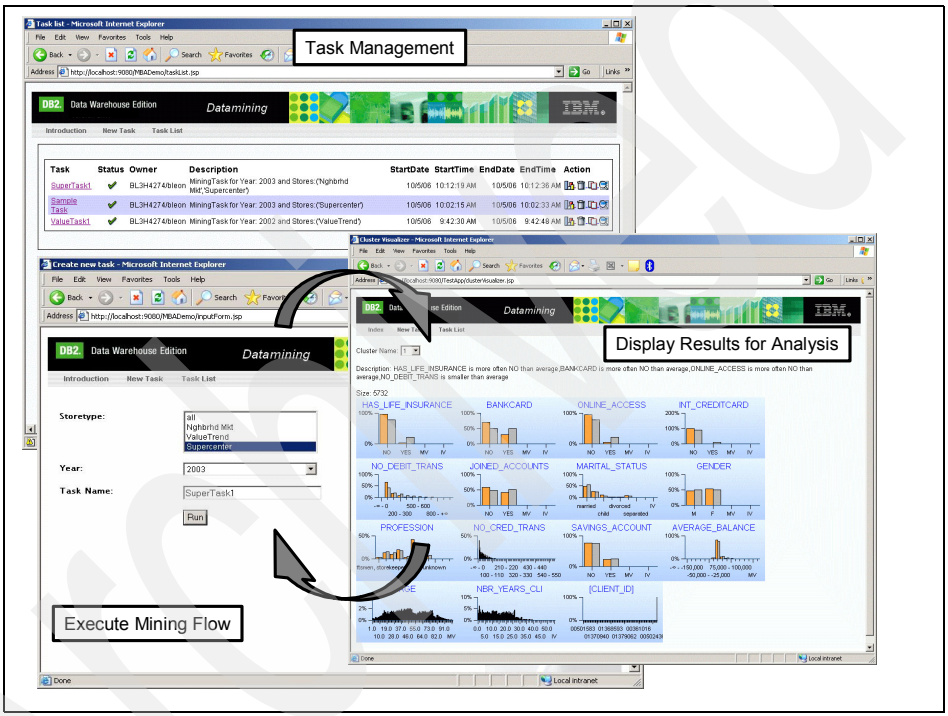


Figure 8-5 Using a Miningblox application

A Miningblox application consists of several components, as shown in the left side of Figure 8-6. There is the DB2 Warehouse Administration Console, which provides the runtime support for executing the mining flows that were developed in the Design Studio and deployed as a Warehouse Application. There is also the Miningblox application server components installed as part of the Alphablox runtime server. Finally, there is the Miningblox tags that are embedded in a jsp page that provides the user interface as well as the logic to invoke the data mining runs. On the right side of Figure 8-6, the user, through a browser, opens a Web page for the Miningblox application, which contains Alphablox tags as well as the Miningblox tag extensions. The Miningblox Web application invoked functionality in both the Alphablox server as well as the DB2W Administration Console to accomplish the work and display the results back to the user's browser.

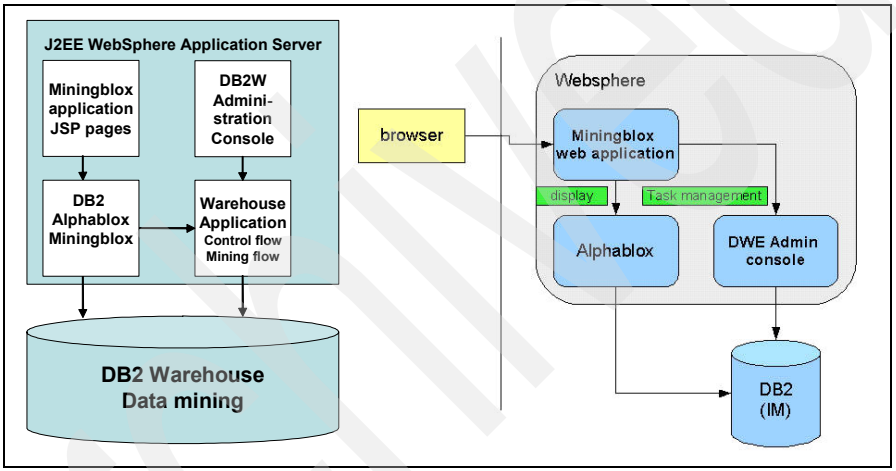


Figure 8-6 Components of a Miningblox application

Figure 8-7 lists the Miningblox tags that are available in addition to the wealth of tags available with Alphablox. There are visual component tags for including on data input forms and for displaying mining results. There are also tags for executing and administering data mining tasks. Refer to the *IBM DWE Miningblox: Administration and Programming Guide*, SC18-9805 for more information about using these tags. The same tags are utilized in custom developed applications and also used by the Miningblox Application Wizard.

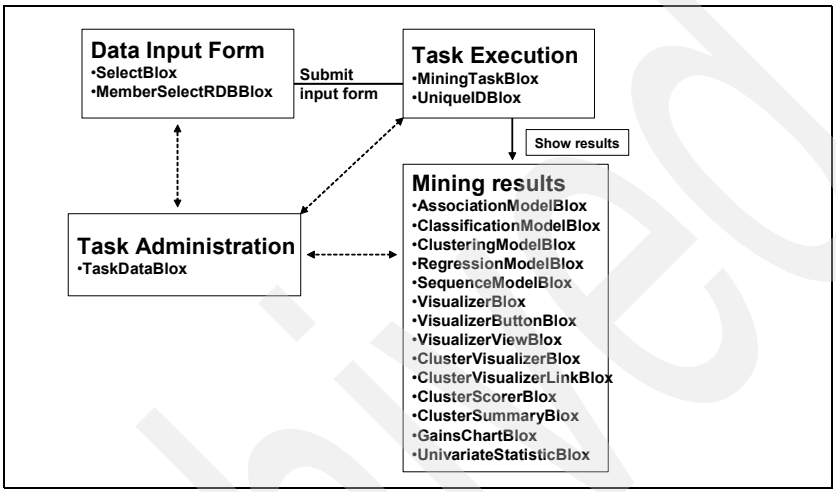


Figure 8-7 Miningblox tags

Using Miningblox can help accelerate the delivery of sophisticated mining applications to line-of-business users. Further details about Miningblox examples can be found in Chapter 5, “DB2 Warehouse tooling for data mining” on page 51.

IBM Business Partner applications

In addition to the IBM tools found in DB2 Warehouse, many IBM Business Partners, such as Business Objects and Cognos, use these same Application Programming Interfaces (APIs) to provide data mining applications and templates. These are usually specific industry type of applications targeted to the business user that would provide capabilities such as Market Basket Analysis.

Other applications

Of course, there is always the capability to use normal programming languages to develop applications. In a lot of circumstances for real-time data mining integration, this is the only choice. Any language that can call DB2 routines can make use of the any of the SQL-based data mining calls. Java beans can also be generated for Java applications. See 8.3, “Techniques for embedding data mining solutions” on page 227 for more information about the Application Programming Interface for DB2 Warehouse Data Mining.

8.3 Techniques for embedding data mining solutions

Having data mining functions embedded directly into the DB2 Database Server engine opens up a number of possible entry points to creating and deploying data mining solutions. DB2 Warehouse 9 contains tooling components to allow the rapid embedding of data mining analysis in Business Intelligence, eCommerce, user reporting, and even back into Online Transaction Processing (OLTP) systems. In addition to IBM tooling, other tools and languages can use many of the provided interfaces for accessing and utilizing the embedded data mining functions of DB2 Warehouse 9.

At this point, it must be understood that data mining is actually happening inside of the DB2 Database engine. Mining models and data are stored in relational tables while the data mining is accomplished using SQL functions. Referring to Figure 8-8, when a data mining run is executed to create or update a data mining model, this is accomplished by using one of the various ways to call data mining functions. Everything is done within DB2. The data is stored in DB2 tables, the called functions are DB2 functions, and are executed by the DB2 engine. During execution, the data does not flow outside of the database engine, which is important in terms of performance. Finally, the mining model itself is created and stored in a DB2 relational table.

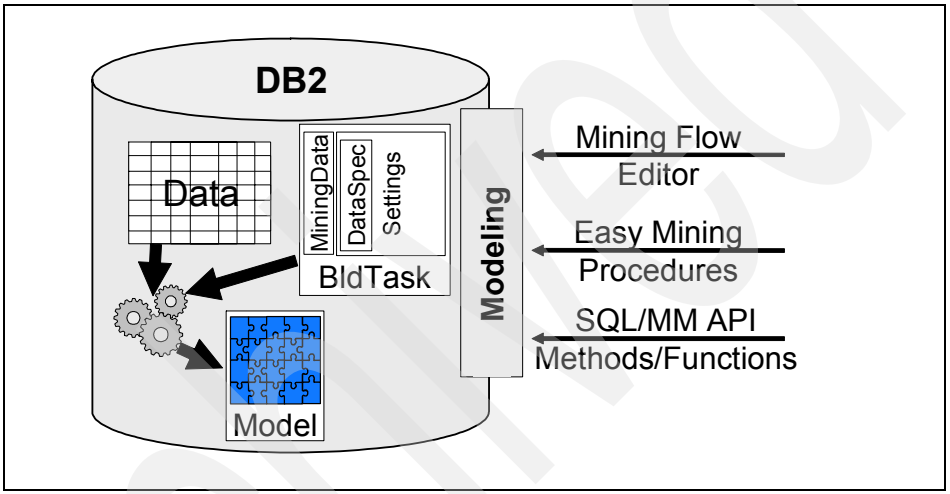


Figure 8-8 Mining runs in the database

When the DB2 Warehouse 9 Visualizer is used to graphically explore the mining model, seen in Figure 8-9, the model is extracted from the relational table and displayed graphically.

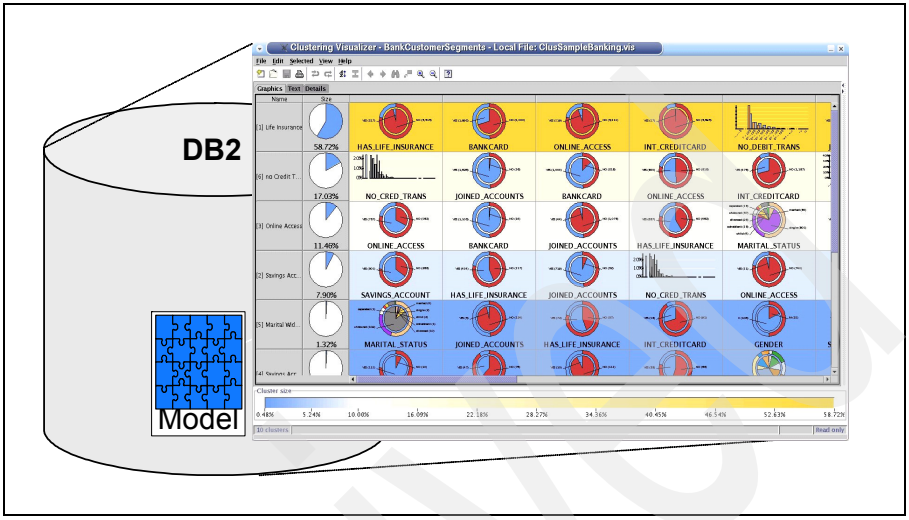


Figure 8-9 Exploring a mining model from the database

When a model is applied to new data, this is again all accomplished within the DB2 engine and the results of the scoring are stored back in relational tables. These results can then be used by any DB2 application or even fed into other analytic structures like relational cubes, as shown in Figure 8-10 on page 230.

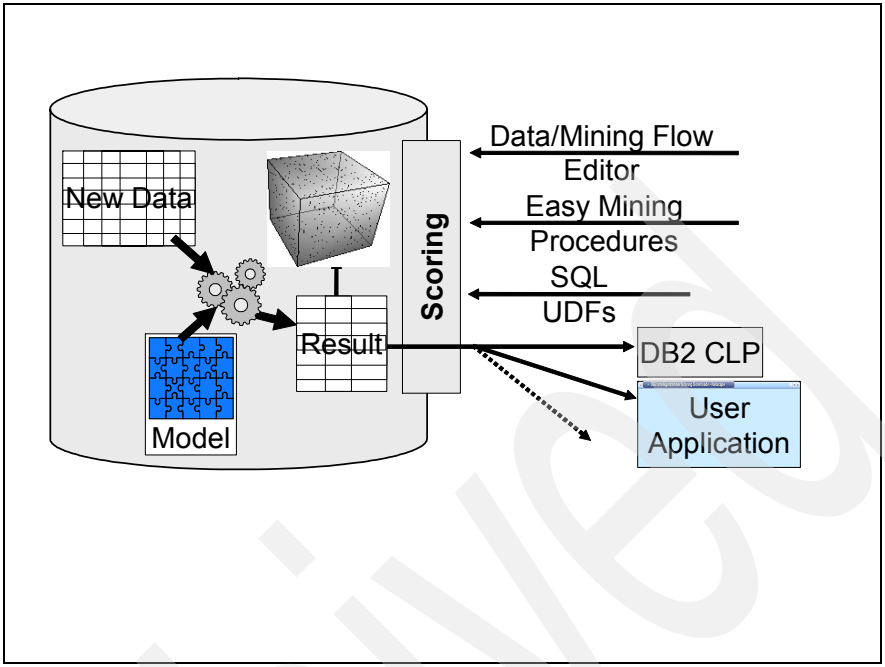


Figure 8-10 Scoring in the database

Figure 8-11 on page 231 shows various access points for invoking data mining functions from tools such as the DB2 Warehouse 9 Design Studio, Excel®, or any business application. In fact, *any tool*, such as reporting tools, that can call this SQL API can utilize data mining.

The basic application programming interface (API) for embedding data mining functionality into applications is SQL, as shown in Figure 8-11 on page 231. This API is realized as DB2 extensions to the SQL language in the form of user-defined functions (UDFs), user-defined data types or user-defined types (UDTs), user-defined methods (UDMs), and stored procedures that are installed when a DB2 database is enabled for data mining.

The SQL data mining API provides multiple levels of abstraction that allows calls at differing levels of granularity. The lowest level is called the SQL/MM API from which one can compose mining tasks that are hand-tailored for individual requirements. This requires fairly deep skills. The higher level of abstraction is the Easy Mining Procedures API. This is a task-oriented API consisting of stored procedures with simpler syntax. This API concentrates on the more common mining tasks. With both levels of API, one can create data models and apply data models using scoring.

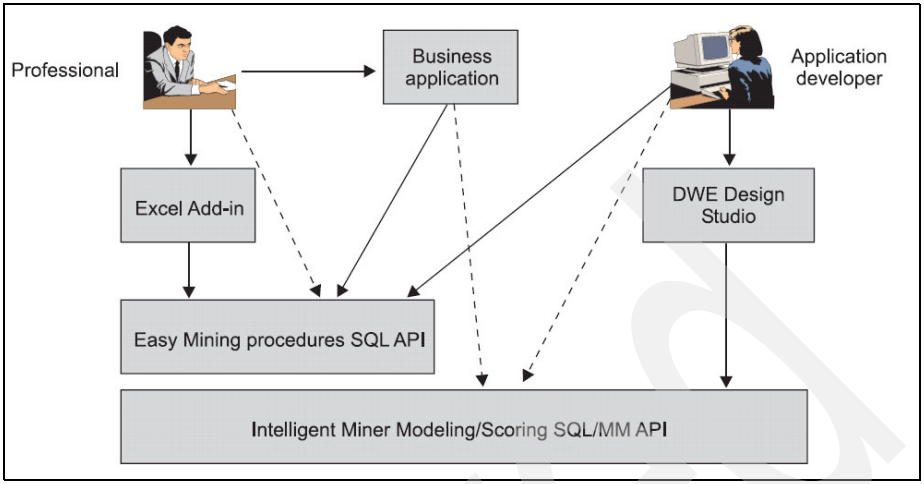


Figure 8-11 Using DB2 Warehouse 9 data mining in the business processes

Above the API level are the applications and tools that use the SQL functions to incorporate data mining tasks into whatever business process they are implementing. Custom-developed business applications can certainly be developed using either the task oriented API of Easy Mining or the more powerful SQL/MM API using any language that supports JDBC™, CLI, ODBC, or SQLJ interfaces.

There are also many tools that can use the SQL API and, as such, can invoke data mining functions. The most simple of these tools is the DB2 command line, where the SQL calls can be made in an interactive environment. As an extension of this, the data mining SQL can also be used in DB2 scripts. Many report tools also have the ability to call the data mining SQL functions. In addition, there is an Excel plug-in to do the same. This plug-in is available from IBM alphaWorks, `http://www.ibm.com/alphaworks`, which also uses this SQL API through Microsoft®'s ActiveX® Data Objects and the IBM OLE DB provider for DB2. The plug-in can display the SQL code and it can be grabbed for use in other applications.

There are more sophisticated tools developed to make working with data mining very easy, such as the DB2 Warehouse 9 Design Studio. See 5.2, "Design Studio" on page 53 for an overview of the DB2 Warehouse 9 Design Studio. Using the Design Studio, data mining solutions are developed using the graphical flow editors and then can be deployed to the Administration Console runtime environment for execution. Based on the graphical elements in the Design Studio, the SQL/MM level of the API is used to implement the solution. From mining flows in the Design Studio SQL code or Java Beans can be generated for reuse in other applications.

Comparing the APIs

In this section, we compare the SQL/MM and Easy Mining APIs. We highlight their differences and compare them with the Design Studio's graphical user interface.

SQL/MM

The SQL/MM API is the base, lowest-level API for DB2 Warehouse Data Mining. All of the other APIs and tools are based on this API. All other APIs and tools are higher level abstractions of this API. This API exploits object-relational extensions to DB2, such that objects are implemented as user-distinct types and user-defined structured types that are stored in large object columns (LOB) in relational tables. Methods in the form of user-defined methods (UDM) and user-defined functions (UDF) are invoked against the object. Using these objects, methods, and functions enable data mining users to use the SQL language to perform data mining functions.

Example 8-1 shows the SQL/MM code that creates an associations model using input from the table RETAIL. Within the three SQL statements table objects, user-defined types and user-defined methods specific to data mining are utilized. While this allows full control over data usage, mining task setting, memory usage, working directory, mining algorithm, and tuning parameters, it is still a bit complex and requires a good understanding of data mining. Contrast this with the Easy Mining calls in the next section.

Example 8-1 Creating a mining model using the SQL/MM API

```
INSERT INTO IDMMX.RULETASKS(ID,TASK)
WITH MYDATA(MYMININGDATA) AS (
VALUES(IDMMX.DM_MiningData()..DM_defMiningData('RETAIL'))
)
SELECT 'RetailRulesTask',
IDMMX.DM_RuleBldTask()..DM_defRuleBldTask(
MYMININGDATA,
IDMMX.DM_RuleSettings()
..DM_useRuleDataSpec(MyMiningData..DM_genDataSpec()
..DM_setItemFld('ITEMID')
..DM_setGroup('TRANSLATE')
..DM_setRuleFilter(IDMMX.DM_RuleFilter()
..DM_setMaxNumRules(100)
..DM_addRangeConstr('support',13,100,'isInClosed')
..DM_setItemConstr('Cream','isInHead',1)
..DM_setItemConstr('Milkprod','categoryNotInBody',2)
)
..DM_setPowerOptions('-buf 1024')
..DM_setAlgorithm('A-Priori',<NumBins>10</NumBins>')
```

```
)
FROM MYDATA;

CALL.IDMMX.DM_buildRuleModelCmd(
'IDMMX.RULETASKS', 'TASK','ID','RetailrulesTask',
'IDMMX.RULEMODELS','MODEL','MODELNAME','RetailModel1');
```

Easy Mining

Easy Mining is the API most likely to be used in an application. It is a set of stored procedure calls that simplifies the invocation of data mining within the SQL language. The calls are group into three categories:

- Typical mining tasks
- Basic mining tasks
- Preprocessing/utilities

The Easy Mining procedures for typical mining tasks are the easiest to use and can solve most of the typical business problems in various application areas, such as banking or manufacturing, without having in-depth mining skills. They are easy to use because only one procedure call is required versus several SQL statements. The Easy Mining stored procedures have a simplified parameter list with most parameters set to appropriate defaults. As seen in Figure 8-12, the Easy Mining tasks correspond to the main steps of the data mining process.

Mining tasks	Easy Mining procedure
Finding deviations	FindDeviations
Finding groups with similar characteristics	ClusterTable
Finding relationships	FindRules
Finding sequential rules	FindSeqRules
Predicting future behavior	PredictColumn
Predicting an outcome	PredictColValue
Finding explanations for specific events	ExplainColValue
Finding most important fields	FindMostImpFields

```
Call IDMMX.BuildClusModel ('BankingClusEasyModel', 'BANKCUSOMERS')
```

Easy Mining stored procedure to build a Clustering model

Model name

Data source

Easy Mining typical procedure call to create a mining model

Easy Mining stored procedure to score data against a Clustering model

model name

data source

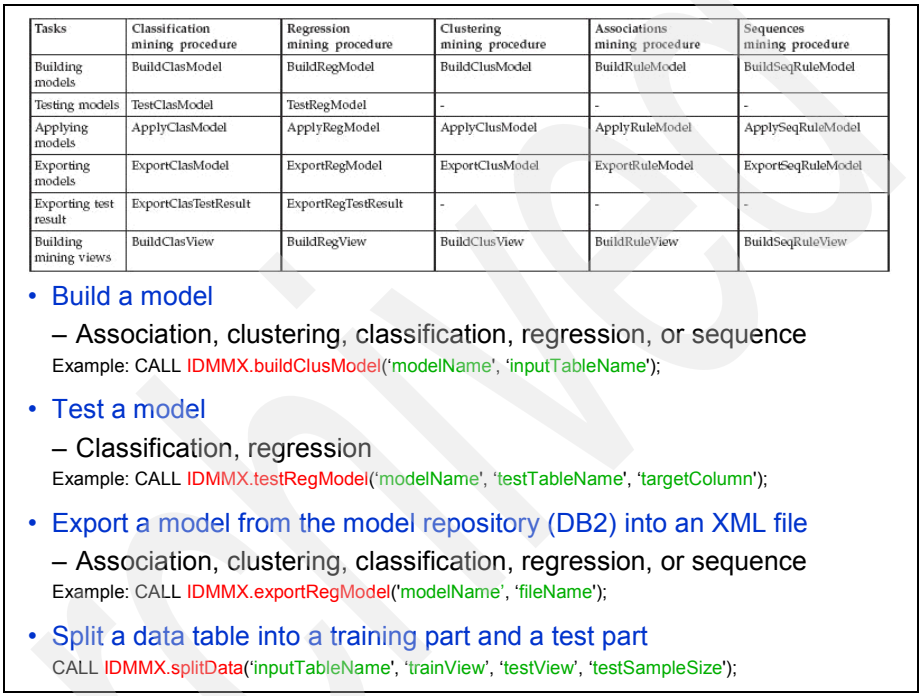
```
Call IDMMX.ApplyClusModel ('BankingClusEasyModel', 'BANKING_SCORING',
'BANKING_RESULT','CLUSTERID','QUALITY','CONFIDENCE')
```

View name and its columns to store the scoring result

Easy Mining typical procedure call to score a mining model

Figure 8-12 Easy mining typical tasks and examples

The basic group of Easy Mining stored procedures are at a lower level and correspond to the SQL/MM API, but are easier to use as the syntax has been simplified and focuses on the more frequently used concepts of SQL/MM. Unlike the typical group of Easy Mining procedures, the parameters can be modified if the user has the required data mining knowledge. Figure 8-13 is a list of the Easy Mining basic tasks with some usage examples. Compare these calls to the Easy Mining typical tasks in Figure 8-13 and the SQL/MM API calls in Example 8-1 on page 232.



Tasks	Classification mining procedure	Regression mining procedure	Clustering mining procedure	Associations mining procedure	Sequences mining procedure
Building models	BuildClasModel	BuildRegModel	BuildClusModel	BuildRuleModel	BuildSeqRuleModel
Testing models	TestClasModel	TestRegModel	-	-	-
Applying models	ApplyClasModel	ApplyRegModel	ApplyClusModel	ApplyRuleModel	ApplySeqRuleModel
Exporting models	ExportClasModel	ExportRegModel	ExportClusModel	ExportRuleModel	ExportSeqRuleModel
Exporting test result	ExportClasTestResult	ExportRegTestResult	-	-	-
Building mining views	BuildClasView	BuildRegView	BuildClusView	BuildRuleView	BuildSeqRuleView

Figure 8-13 Easy mining basic tasks and examples

There is a set of Easy Mining procedures for other tasks to help prepare the data for an Easy Mining procedure to accomplish administrative tasks such as handling error messages, canceling Easy Mining procedures, or working with the trace file. The following preprocessing and utility procedures are available:

- SplitData
- GetLastError
- SetTraceFile
- GetTraceFile
- GetCancelTask
- GetCleanUpTask

Design Studio

While not an API, the DB2 Warehouse Design Studio provide a graphical user interface over the data mining APIs using a graphical flow editor. In fact, a mining flow has abilities beyond just the ability to invoke the data mining functions of the API, particularly in the area of preparing the data for mining using preprocessing operators, as shown in Figure 8-14. Data mining applications can be built graphically and executed within the Design Studio. However, implementing data mining in the Design Studio offers a lot more capability. Data Mining flows can be embedded into Control Flows and integrated with other types of Design Studio data flow and other processing functions. In addition, a data flow has a subset of data mining operators so that a model can be refreshed or scored within a data flow. These warehouse applications developed in the Design Studio can be deployed to a runtime server for production scheduling. Miningblox applications can also be built to use these mining flows in a user Web-based application. See Chapter 5, “DB2 Warehouse tooling for data mining” on page 51 for more details on using the Design Studio, and Miningblox.

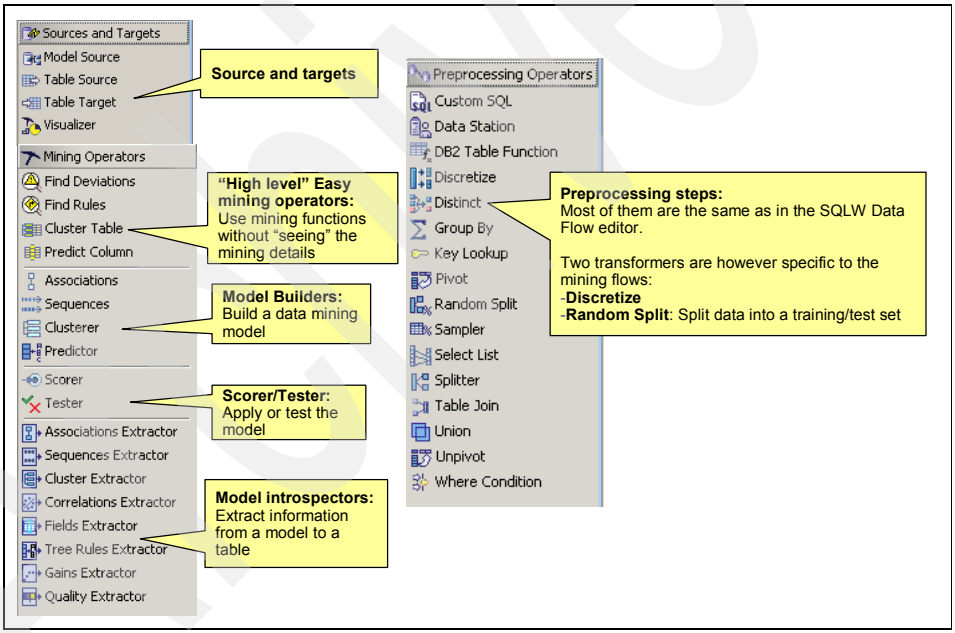


Figure 8-14 Data Mining functions in the DB2 Warehouse Design Studio

9

Solving a business problem with data mining

In this chapter, we discuss the important aspects of implementing a data mining solution. The whole chapter is aimed at the business and data analyst, and the business intelligence professionals that support them. A case study scenario is used to illustrate the key steps involved in the overall project including definition of business objectives, the data mining process, analysis, and interpretation of results. The case study is based on a retail department store, although a similar process could be used to implement data mining solutions in different industry scenarios.

The focus of this chapter is on the business aspects of the process rather than the technical aspects of the implementation. Technical details are discussed in others chapters of this book.

In this chapter, we also describe the process of solving a business problem using a data mining solution. The components of this process include the following:

- Data mining solution overview
- Process to implement a data mining solution
- Steps in implementing a data mining solution
- A case study for a retail department store

9.1 Data mining solution overview

A data mining solution consists of applying data mining techniques against business data to obtain business insight that can be used to improve business operations and performance. The results can be used to identify new business opportunities, improve the existing business process, solve business problems, and support the business decision process. An overview of the main elements of a data mining solution is shown in Figure 9-1 on page 239.

Business problems and business opportunities are the drivers for data mining solutions. The business problem defines the business requirements for the data mining project. Business requirements are represented by variables that can be modeled in a data mining process. And then the data mining process is executed against the source data to discover new information and give insight about a business problem or opportunity.

The source data consists of transactional data and other data structures that are used as input by the data mining process. Depending on the business objectives and nature of the existing data, additional data elements may have to be created or transformations of existing data may be required.

The data mining process consists of data mining models, algorithms, and techniques that are executed against the data. The data mining process then produces results that are interpreted to gain business insights.

Business insights are achieved by analyzing the data mining results. The analysis consists of evaluating the data mining results in the context of the business problem. And business insights are translated into actions to be implemented to reconcile or eliminate the business problem.

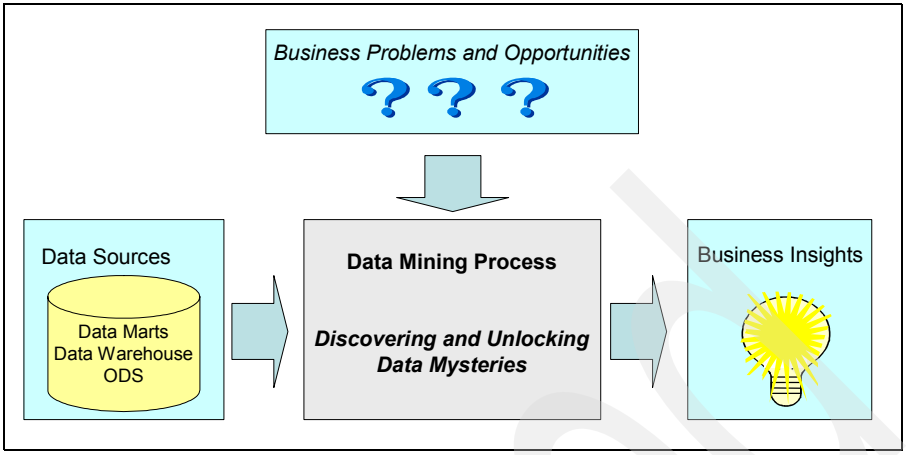


Figure 9-1 Data mining solution overview

9.1.1 Process to implement a data mining solution

In general, the primary focus when implementing a data mining solution is on the mining and discovery aspects of the process. Data mining, however, is just one of the steps in the overall process. Figure 9-2 on page 240 illustrates the major steps and activities in the data mining implementation in the sequence in which they occur.

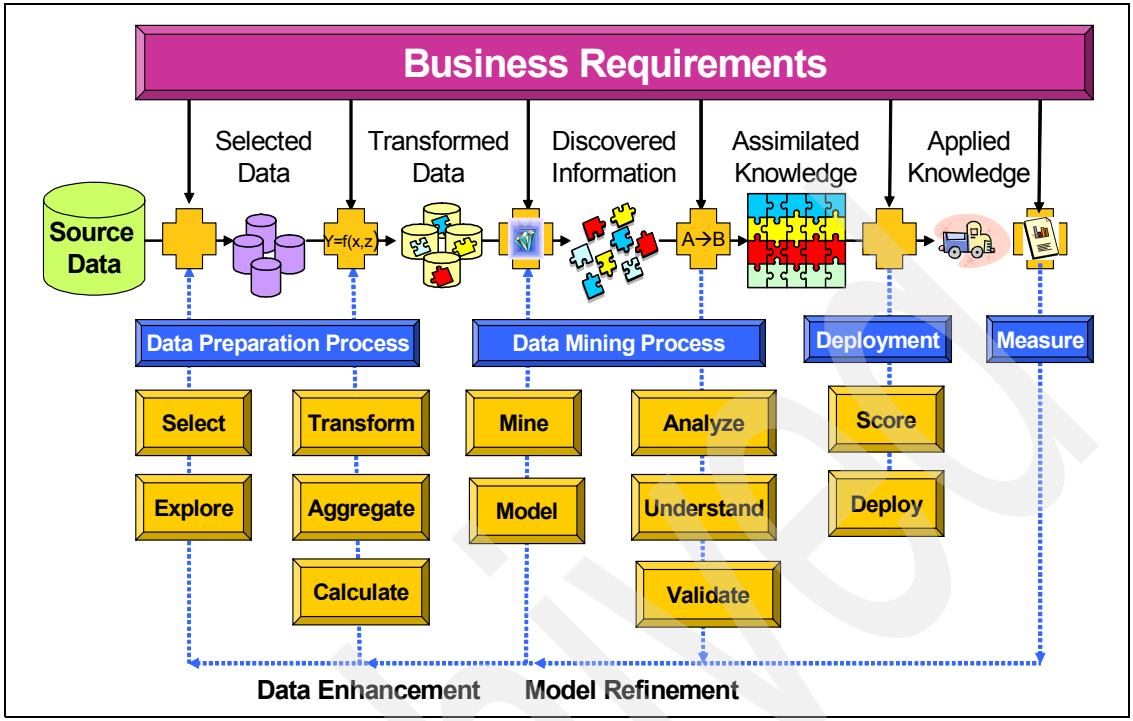


Figure 9-2 Data mining solution process

The business requirements drive the entire data mining solution. They are the basis on which to establish the initial project plan, analytical approach, and activities. They guide the project team throughout all steps and, finally, serve as the measuring stick for judging project results against business expectations.

The data preparation process involves all of the data manipulation tasks. Activities include identifying internal and external data sources, selecting data sets (or portions thereof) to be used in the mining process, defining analytical variables for the mining models, defining transformation rules for existing data, creating new variables, data aggregation, and calculations.

The data mining process includes the creation and execution of data mining models. This process generates results used to validate and refine the models and to provide business insights.

Although the steps in Figure 9-2 on page 240 are performed in the sequence shown, in practice the overall process is highly iterative, particularly the data preparation and data mining processes. Typically, many loopbacks over one or more steps are required to refine the data or the models. Additional data preparation may be needed to modify variables or create new ones. Model parameters and settings may also require adjustment to refine the models.

The deployment process involves activities to incorporate data mining results into the business process. Depending on the complexity of the problem, this activity could require significant resources and planning.

After data mining results have been applied to the business, there should be some measurement to determine if results are meeting the goals and expectations. This assessment may consist, for example, of evaluating the success of a promotion, the effectiveness of treatment protocol in reducing the incidence of a disease, or cost savings from changing a manufacturing process.

There is no simple formula to estimate the amount of effort required to perform a data mining project. Many different factors can impact the amount of effort required to perform any or all of the activities. For example, data preparation could require extra time if a large number of analytical variables have to be designed and created. The execution of mining algorithms could take several hours or just a few minutes, depending on the number of analytical variables involved in the model, the amount of data processed, hardware resources available (CPU, Memory, and I/O), and system workload. Of all the major activities in a data mining project, data preparation typically requires the most effort and can be expected to consume anywhere from 40 to 80 percent of the overall project time, or even more if the data are not already accessible in a data warehouse.

A chart depicting the typical level of effort for each major group of activities in a data mining project is presented in Figure 9-3. These relative values are provided here just as a general guide and will vary from project to project. This chart conveys the idea that a data mining project involves more than just the creation and execution of data mining models, and all of the activities should be taken into account when defining the project plan and scoping the resources to perform the project.

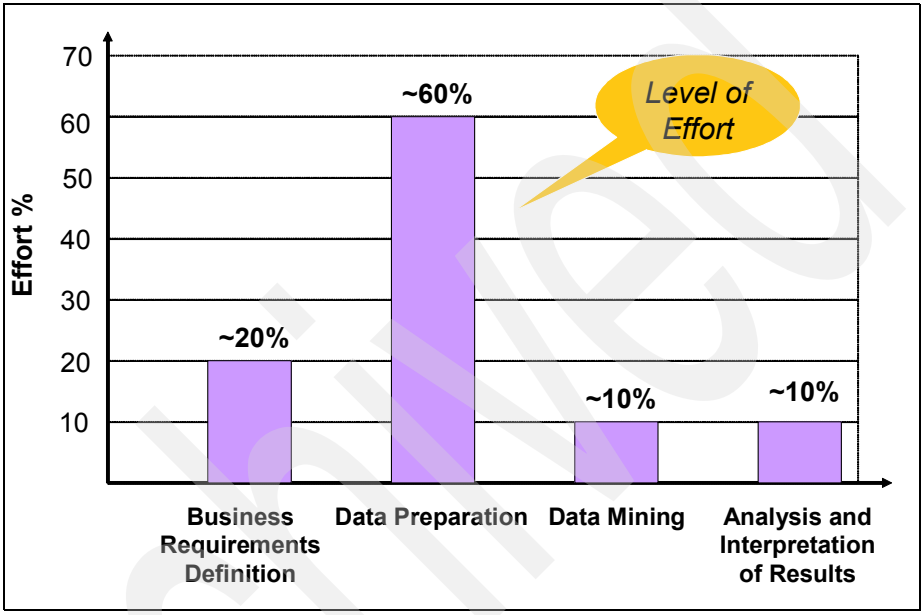


Figure 9-3 Typical level of effort for activities in a data mining project

9.2 Steps to implement a data mining solution

In this section, the steps required to implement a data mining solution are described. These steps are executed sequentially, although multiple iterations are generally necessary in some of the steps. At this point, the business objectives have been defined and validated as suitable for being addressed by a data mining solution.

9.2.1 Step 1: Organize the project team

Despite the need for a good project plan like any other IT project, a data mining project requires special consideration when defining and allocating resources for the project. Data mining projects are implemented to help corporations discover unknown information or gain new insights about the business. Results can be applied at any of the different levels of the decision process: operational, tactical, and strategic. For a successful implementation, some level of executive sponsorship is crucial to guarantee that the project is aligned with strategic goals and that project results will be implemented. Data mining experts and IT resources are fundamental to the project, but, additionally, participation by resources from all of the involved business areas, for example, business analysts, operational managers, and line-of-business (LOB) subject matter experts, is key to the success of the project. This participation is important during different phases of the project, including the initial planning phase to define the scope of the business problem to be solved, the data mining analysis and results validation phase to provide business insights, and the deployment phase to assure that the project results are meeting business expectations and are achieving the desired goals.

A data mining project without executive business sponsorship and without involvement of key elements of the business areas has a high probability of failure, and any results are unlikely to be applied to the business.

9.2.2 Step 2: Understand the business objectives

The business objectives should be clearly defined and understood by the project team before any subsequent project activities are begun. The business objectives of the project should represent a high-value outcome and should have executive support to ensure a political desire to implement the solution. These requirements could be the identification of new business opportunities, the improvement of an existing business process, or a solution to a business problem. Figure 9-4 shows some examples of typical problems that are suitable for a data mining solution.

Industry	Typical Business Problems and Opportunities
Retail	• Improve promotional effectiveness –Cross-sell and up-sell –Determine next-likely-purchase, Market Basket Analysis • Improve store performance –Store profiling and Store layouts
Financial Services	• Improve customer loyalty • Improve customer profitability • Reduce customer attrition • Fraud detection
Healthcare	• Disease management • Patient profiling • Drug interactions, efficacy • Cohort selection for clinical trials • Fraud detection
Insurance	• Fraud detection • Improve customer profitability – Product/service cross-selling, up-selling • Improve customer loyalty
Manufacturing	• Part failure prediction • Warranty claims mitigation • Inventory management and replenishment

Figure 9-4 Typical problems and opportunities in data mining projects

A data mining project without clearly-defined objectives and executive support has a high potential for failure. Not only will time and resources be wasted in such a situation, but any future adoption of data mining technology inside the corporation may be jeopardized as well.

In general, defining and understanding the business objectives for a data mining project is not an easy and straightforward task. It usually requires the participation and collaboration of multiple members of the organization, including business analysts with domain expertise in the related business areas, data analysts who can identify data gaps based on existing data and requirements for the project, and data mining experts who can define the best analytical approach to address the business requirements and solve the business problem.

Definition and understanding of business requirements is one of the most important activities in a data mining project. All of the effort to define and understand the business requirements not only sets the stage for a successful project, it also minimizes rework and project delays. All other activities are based on the requirements defined at the beginning of the project. If the requirements are not properly defined or understood, the project results inevitably will not meet expectations, and significant rework will be required to realign the project to achieve the expected results.

It is also important to clearly define the expectations of the project. A project with invalid expectations could result in project failure and executive dissatisfaction. For example, consider a data mining project with the goal of improving the response rate for a campaign. There are variations among different campaigns, but expectations for incremental, single-digit percentage improvement would be acceptable and typically achievable. Expectations for double-digit percentage improvement, however, may be difficult to achieve and could cause frustration and dissatisfaction from LOB and executive sponsors.

Project expectations vary based on business requirements, but a good way to measure project results is to define metrics and expected improvements, for the performance indicators. Later in the process, when the project is being implemented, these metrics can be used to calculate and measure improvements. Measures such as average sales per week, average transactions per item, response rates to a campaign, churn rates, and out-of-stock rates are just a few examples of metrics that could be used to measure project results.

9.2.3 Step 3: Define the data mining approach

This activity involves the determination of the most appropriate data mining methods and algorithms to be used to solve the business problem. It requires not only expertise in data mining and the data mining tools being used, but also requires a very good understanding of the business requirements.

A data mining solution to solve a given business problem may require a combination of different data mining algorithms or methods. Figure 9-5 shows examples of typical business questions associated with a business problem, and a typical data mining approach.

A given business problem could also be addressed using different approaches. The results may be very similar or could vary significantly, depending on the data mining methods used. A data mining specialist could decide to use different approaches and compare results, then apply the method that provides the best business results.

Typical Business Questions	Data Mining Approach
What do my customers look like? Which customers should I target in a promotion? How can I identify high-potential prospects (lead generation)?	Clustering
Which products should I use for the promotion?	Associations + Sequential patterns
How should I layout my new stores? How can I improve customer loyalty? How can I detect a potential fraud	Clustering + Associations
What is the likelihood of a part failure? When one part fails, what other part(s) are most likely to fail soon?	Regression
Which products should I replenish for a promotion?	Associations
Which of my customers are most likely to churn? Who is most likely to have another heart attack?	Classification or Regression
What item is a customer likely to purchase next?	Sequential Patterns

Figure 9-5 Defining the data mining approach

9.2.4 Step 4: Prepare data for data mining

In this section, we describe the steps for preparing the data for data mining.

Step 4-1: Data selection

This activity consists of identifying the available data sources and extracting the data that is needed in furthers steps of data preparation. This step includes determining what data is available to support the data mining project. The data selection varies with the business objectives and data requirements for data mining. Source data could be extracted from data warehouses, operational data stores, data marts, or transactional systems. External data such as demographic information, economics, weather, and so forth, also can be used to complement the information available internally.

Another consideration in data selection is what data should be used to support the data mining process, based on the business requirements. For example, if a retailer wants to determine the best set of product assortments to be replenished for a Christmas promotion, it would be better to perform the data mining analysis using historical transactions that are Christmas-related.

Step 4-2: Defining analytical variables

Defining what variables should be created for the data mining model is one of the most important tasks in the overall data mining project. This requires knowledge of existing data structures, understanding of the business requirements, and understanding of the data requirements for the data mining process. The variables (attributes) used in the data mining models are commonly called analytical variables, which vary based on business objectives.

Definition and creation of analytical variables requires collaboration among the members of the data mining project team. Some initial data investigation is required to determine what data is available and the data gaps to support the data mining application.

Typically, the existing data contains only the base variables, such as total customer spending amount, net revenue, purchase quantities, and so forth. In some scenarios, the data may be in a detailed transactional format, such as Point-of-Sale (POS) or account details. Regardless of the source, data needs a certain amount of data preparation to support the data mining needs and that is because data mining models requires specific data layouts and data formats. For details about data layouts for data mining, see 6.2.2, “Data requirements for the mining methods” on page 99.

For example, in a retail store, a customer's spending behavior is represented by several variables, including the spending pattern of that customer among different departments. Representing the spending pattern for a customer involves calculating the *percentage spent in each department*. In addition, a count of the number of transactions per customer can be calculated.

In a bank, creating a set of variables to represent customer behavior may include calculating such variables as *number of returned checks*, *number of ATM debits*, *number of Web transactions*, *total investments*, and *percentage balances* (for a set of accounts such as checking, saving, and IRA).

The following are examples of variables that are commonly used in marketing applications and can be easy calculated from existing variables:

- Average amount spent over a certain period of time
- Average number of transactions over a certain period of time
- Average number of items per transactions over a certain period of time
- Count of distinct of items purchased over a certain period of time
- Count of distinct departments, stores, and so forth, in which the transactions were performed over a certain period of time
- Number of total transactions performed over a certain period of time
- Total spent over a certain period of time
- Total quantity spent over a certain period of time
- Total cost of amount returned over a certain period of time
- Total quantity returned over a certain period of time
- Percent spent on specific departments, products, product categories, and so forth, over a certain period of time
- Percent spent each week of the year, month of the year, day of the week, and so on
- Percent of total spent in a certain period of time versus the available credit in the same period
- Highest price paid for a certain product over a certain period of time
- Percent discount over a certain period of time
- Percent of spending by sales channels (internet, direct sales, store)
- Ratio of price paid for most expensive item to least expensive item
- Ratio of total debit to available line of credit

Step 4-4: Data exploration

After identifying the analytical variables and before performing any transformations or calculations, there is still a need for further investigation of the source data. The source data used in a data mining project may be collected from numerous sources, and its quality and content may vary considerably. Understanding the quality of the source data is very important for a successful data mining project. Each variable needs to be evaluated to determine whether it brings any potential value to the analysis. All of the data issues need to be understood and resolved before proceeding with data mining. The understanding of the data gained in this step is used in defining the transformation rules when calculating new analytical variables.

DB2 Warehouse Design Studio provides capabilities for univariate, bivariate, and multivariate analysis that helps data analysts to quickly perform analysis to identify data quality issues. See Chapter 6, “Data preparation for data mining” on page 97 for a more detailed discussion on this topic.

Step 4-5: Perform the ETL process

This step consists of performing the transformations, calculations, and derivations required to generate the analytical variables and format the data as required for the data mining models. Complex calculations and formulae should be documented to help in defining and refining the ETL process. This documentation can also be used to validate calculations after the ETL process has been completed.

The ETL process results in table(s) containing variables for the data mining process. After the ETL process has been completed, it is important to validate the contents of the resulting table(s) and ensure that the calculations are correct. Details on how to perform data preparation tasks are described in Chapter 6, “Data preparation for data mining” on page 97.

9.2.5 Step 5: Perform the data mining process

The objective of this step is to apply data mining algorithms to the prepared data. For clarification in this discussion, analysis and interpretation of results are presented as separate steps; however, this is an iterative process and, in practice, analysis and interpretation occur together.

At this point of the project, it is assumed that a data mining approach has already been selected. Now, a data mining expert has to define the data sampling approach, treatment of variables properties (for example, active or supplementary), and model the properties and parameters.

Step 5-1: Define the sampling approach

Sampling may or may not be necessary when performing a data mining process. In a situation where the source data volume is large, it may be desirable to use only a portion of the data during model development, for better performance in execution times. In other cases, the analyst may want to focus on a particular time period or cross section of a population and, thus, needs to extract the data relevant to the analysis. Whenever a subset is sampled from a source, it is essential that a random sampling process be used. Sampling schemes such as "every tenth record" or "the first 10,000 rows" are practically guaranteed to generate misleading and invalid mining results because such schemes do not ensure a subset that is representative of the parent data source or population.

In the context of predictive modeling, training and testing sets are used for model building and validation, respectively. These sets are constructed by randomly extracting mutually exclusive, distributionally similar sets from the parent source.

DB2W Design Studio offers a variety of functionality for data sampling, either for creating a data subset or for generating training and testing sets. See 7.2.1, "Building a clustering mining flow" on page 142, 7.3.4, "Sampling transactions" on page 175, 7.4.4, "Sampling transaction groups" on page 185, and 7.7, "Oversampling to create an enriched training set" on page 206 for more discussion of sampling logic and techniques in Design Studio.

Step 5-2: Define data mining models

When defining the data mining models, the data mining analyst needs to define what variables from the source table are to be selected and included in the mining models. For each variable, the analyst needs to define whether it should be active, supplementary, inactive, or system-determined. *Active* means that the variable is used in building the model. *Supplementary* means that the variable is not used in constructing the model, but it is displayed in the results. *Inactive* means that the variable is excluded from the model. *System-determined* means that the mining algorithm will determine whether to treat the variable as active or supplementary. The variable will be active unless each record has a unique value (such as a key) or is highly correlated with another variable that is active (to avoid double-counting the underlying effect represented by the two correlated variables).

Defining variable properties is part of the iterative approach in the data mining process. When analyzing the data mining results, a data mining analyst may decide to switch variable settings from supplementary to active or vice versa, re-execute the data mining process, and then compare results.

In addition to defining variable properties, a data mining analyst also defines the appropriate parameters for each of the data mining procedures being used. These parameters affect the overall data mining process by controlling some aspects of how a model is built or by impacting the execution time, thereby impacting the mining results. The settings are specific to each of the data mining procedures. In general, a data mining analyst may choose to start with defaults and modify those parameters based on intermediate mining results and perhaps the analyst's own experience.

Design Studio provides an easy-to-use interface to create mining flows. These mining flows can generate data mining models, write results to output tables, and display visualizations of the mining results.

Step 5-3: Execute data mining models

Execution times for data mining models can be significant, depending on the number and types of variables in the model, the algorithm selected, model parameters, data type, data volume, and system resources. For these reasons, it is difficult to predict *a priori* how long a particular mining model may take to execute.

After a model has been created and applied, the next step is to analyze and interpret the results. If the results are not satisfactory or the quality of the model is not considered reasonable, then a data mining analyst may decide to modify the mining model (and perhaps the data as well) and execute the model again until the quality of the results is satisfactory. These modifications may include changing the composition or treatment of variables used in the model or changing the parameters and settings for the model.

9.2.6 Step 6: Analyze, interpret, and validate mining results

Analysis of the data mining results is one of the most important steps of the entire process. This activity requires the participation of business analysts, data mining experts, and other members of the data mining project. Executive sponsors also should participate in the analysis for clarification and validation of findings in terms of the business objectives.

The activities in this step depend very much on the kind of application that is being developed, but in the first iterations with the model, it is not surprising if the data mining team does not obtain significant insights from the results. The results may point the team to modify the data or model, or even guide them in a different direction. In fact, the team can fully expect this to happen to some extent, but the impact to the project can be minimized by performing the initial steps properly. This underscores the importance of properly understanding the business requirements, defining the mining approach, and preparing the data at the outset of the project. The better that the initial steps are performed, the more quickly the project team can make the necessary changes and proceed.

Significant improvements in terms of model quality and business results can be achieved by adding or removing analytical variables in the model, changing model parameters, or changing mining techniques. For example, when performing clustering (customer segmentation applications), if the model is not producing a reasonable number of clusters with sufficient characteristics to differentiate them well, better results may be achieved by removing common variables from the model and re-executing the segmentation model. The same applies to predictive models. For example, if a single analytical variable has a very high contribution to the model to the exclusion of all the other variables (indicating that it is highly correlated with the dependent variable and, thus, measures very much the same thing), then just removing it from the model could provide completely different and better results.

When developing rules models (associations and sequences), it is very common to re-execute the model using different settings for support and confidence thresholds. For example, if the threshold support level is set too low, these algorithms may generate a huge number of rules. However, many of the rules probably would not be actionable. Equally, if the threshold support value is set too high, only a small number of rules may be discovered, few if any of which are actionable or insightful.

9.2.7 Step 7: Assimilate knowledge for business actions

This step closes the loop in terms of development of the models. The basic activities are to document the findings and translate them into business actions based on the requirements defined at the beginning of the project.

One of the challenges in this step is to present the new findings in a convincing and business-oriented way that can be easily assimilated by executive sponsors. The findings need to be related directly to the business objectives that were set at the project kickoff step. Industry expertise is essential for knowing how to apply the results to accomplish business goals. This activity requires involvement of business analysts, data mining experts, executive sponsors, and others in the LOB.

The business actions depend on the kind of application being developed and the business objectives. Findings can generate multiple business actions. For example, a newly discovered and profitable segment of customers can result in multiple business actions, such as a direct mail campaign, discounts on items that drive sales of other items, and inventory planning to support new demand that would be generated by the campaign. A newly discovered set of associations rules can result in business actions such as development of a marketing campaign to cross-sell new or existing products or services to customers, store layout changes to better leverage shopper preferences, or changes in a Web sales application to up-sell or cross-sell products and services.

9.2.8 Step 8: Deploy the data mining solution

After data mining findings have being translated into business actions, the next step is to apply them to the business process. The business actions depend on the business objectives, and the deployment of a data mining solution entails implementing the business actions into the business processes. This could be just new information to feed an existing process, such as a list of target customers to be used by the marketing campaign design team, or a list of affinity products to be considered by the merchandise team to plan purchases. Other actions may result in a more complex set of activities, such as creating new products and services offerings that may require the involvement of multiple departments and members within the organization.

The most important consideration in this step is to ensure that the implementation of the defined business actions take into consideration the business requirements defined at the beginning of the project. A detailed project plan may be required to guarantee that all actions are implemented as originally planned. Partial implementation can lead the project to failure. For example, assuming a data mining project resulted in a series of business actions including the design of a campaign to promote a plasma TV to a new segment of profitable customers and drive revenue from affinity items. If only a marketing campaign is implemented, but the merchandise group does not supply stores with the right quantity of the plasma TV and its associated items, the results of the campaign could be below expectations because stores are running out of stock of the plasma TV or the affinity items.

In addition to the business initiatives, it is also important to consider the resource requirements for deployment of a data mining solution. DB2 Warehouse provides different options to delivery a data mining solution. In a very simplistic way, the data mining findings can be easily extracted from mining models, and the information can be available for business processes and applications. For example, a list of target customers for a campaign or a list of affinity items for merchandise purchase planning can be easily extracted with a data mining process, and results can be stored in relational tables. For real-time scoring, changes in existing processes and applications may require a detailed project plan and may involve a certain amount of resources. Similarly, if one of the business actions is to implement results in existing applications, such as Web applications, portals, or others, there should be allowances for time and resources required for such an implementation. Chapter 8, "Deploying a data mining solution" on page 213 describes in detail the options available for deploying a solution and how it can be implemented and integrated into a business process.

9.2.9 Step 9: Measure project results

This activity is the measurement process of the project results against the business objectives defined at the beginning of the project. It is important to define key performance metrics to facilitate comparing results achieved in the project to the business objectives. By defining key performance metrics, it is a straightforward task to calculate improvements, which are assimilated and understood by executive sponsors and decision makers. When the business benefits of a data mining project are clearly measurable and visible by project sponsors and decision makers, support and sponsorship to implement additional projects are much easier to obtain.

The key performance metrics are related to the business objectives. For example, a data mining project that has a business goal to improve promotional effectiveness should consider defining performance metrics such as current average items per transaction, campaign response rates, out-of-stock rates, and weekly sales volumes. These metrics could be compared for different periods (prior execution without data mining findings versus new results with data mining findings) to determine improvements. If improvements are not observed, then a detailed analysis and investigation should be performed.

Finding the answers to specific questions represented by the performance metrics is important to determine why project results were not achieved. External factors such as economical and political, weather-related events, competitors, and other factors not under control of the data mining project team can affect the project results.

There are certain internal factors that are not directly associated with the data mining project but can also affect the results. For example, a product quality issue could cause a negative impact in a campaign. Overloading a Web server could cause the entire application to go off-line for long periods of time, causing customer dissatisfaction and consequently impacting the results of a Web promotion.

Certain events directly associated with the data mining project can significantly affect the project outcome. For example, distortions caused by a lack of skills or resources in the implementation process could significantly impact project results or even the ability to bring the project to the point of generating results. Furthermore, business objectives may not have been clearly defined at the outset of the project, or some of the business goals are simply not achievable.

After determining the root cause and reasons why the project results were not successful, further actions can be taken, including re-validating the business objectives or making adjustments in the data mining project team or focus. Measurement of the project results not only provides a way to validate whether or not the project succeeded, but it also may provide insight to identify additional business opportunities. All experience and learning gained from one project can provide a significant contribution for future projects.

9.3 Case study: retail department store

In this section, we present a case study scenario to illustrate the process of applying data mining to solve business problems. The case study uses a retail example, but the methodology and analytical process apply to other industries as well.

9.3.1 Case study overview

The business scenario is a composite of real customer experiences. The company name and characters are fictitious.

Company

Value Trend is a leading retail store chain with operations throughout the U.S. Stores are responsible for 80 percent of company sales. The internet business grew significantly during the past few years and now accounts for 20 percent of overall sales.

Value Trend has a large assortment of products including clothes, sporting goods, home furnishings, computers, and electronics. Value Trend is traditionally known for its product quality and services.

Characters and responsibilities

Vince is a VP of Merchandising and is responsible for the sales and profitability of the products sold by ValueTrend. One of the key performance indicators he monitors are identical sales (also referred to as ID Sales) during a certain time period this year as compared to the previous year. He closely monitors poorly performing items and other associated items in terms of sales and profit with an objective of making recommendations to the VP of Marketing on how to improve upcoming promotions.

Ted is a VP of Marketing and is responsible for providing leadership in the Value Trend target marketplace and customer awareness as part of Value Trend's cross-functional product team process. Ted has overall responsibility for researching the marketplace and customer needs, understanding competitive strategies and formulating Value Trend product feature responses, application directives, and market positioning and sales support collateral. Ted's primary role is to seek out market and customer specific technical opportunities and translate them into actionable engineering projects, sales programs, and marketing communications plans.

Mani is a Marketing Manager and is responsible for the planning and execution of marketing campaigns. One of his many tasks includes analyzing past sales and customer behavior with an objective to improve the net revenue. He also needs to identify the customers to be targeted as part of upcoming advertising campaigns.

Brenda is a buyer. Her role is to determine price points and item volumes across the stores that ValueTrend operates. She reviews the sales of items that are a part of the campaigns, and looks for trends over the last eight weeks. She provides feedback on how to price the items and also determines the required volumes. She optimize pricing and volumes, and perform adjustments to forecast figures.

Simon is a data analyst and has several years of experience in data modeling and ETL tools. He has been one of the key resources in the design and implementation of the business applications, such as operational systems, sales, and forecast analysis applications. In addition to his strong knowledge and experience with data models, he also has acquired a very good business background due to his involvement in the development and design of major applications.

Olivia is a data mining expert with over five years of experience using data mining tools, but she has also had previous experience with relational databases. She has a Ph.D. degree in statistics and a Masters degree in business administration. She has extensive experience in the retail business.

Wendy is a project manager with several years of experience in supervising high-profile, high-budget projects in both IT and business. She has led the implementation of the major IT and business projects at Value Trend.

9.3.2 Business requirements

Value Trend senior executives have identified a need to grow business by at least 12% year-to-year to enhance shareholder value and maintain a leadership position in the marketplace. Key performance indicators over the past few months have not indicated growth rates consistent with the year-to-year target; in fact, the last quarter results were far below expectations. The root cause for this poor performance has already been identified, and business actions have been defined by Value Trend senior management. Strategic actions include a possible expansion into new markets by opening new stores in the U.S., but the most important and immediate objective is to improve current sales volumes.

Value Trend's historically loyal customer base has diminished over time in the face of growing competition. New loyalty programs and promotional campaigns are being designed and implemented. Value Trend's marketing department is now focusing on electronics, and especially high-end television products, as the next major campaign. They have defined specific goals for the electronics campaign, such as (1) identify segments of profitable customers, (2) boost storewide sales, and (3) predict customers' next most likely purchases.

9.3.3 Defining the project team

Ted and Vince decide to allocate Wendy to lead the project effort to develop and implement a solution to support their business goals. Because it is a critical project for Value Trend, they request that Wendy keep them involved in some checkpoint phases over the course of the project to assure that the project is moving towards the business objectives.

Wendy reviews the project objectives and identifies the major components to be part of the solution. Data mining, data analysis, and business expertise skills are fundamental for the success of the project. Wendy allocates key resources to participate in the project, including Olivia and Simon as full time participants because of their expertise respectively in data mining and data analysis. Nani and Brenda are also invited to participate in the project, but their participation would be to support some of the key activities such as understanding and validating the business objectives, definition of analytical variables and metrics, and validation of data mining findings.

9.3.4 Understanding of the business requirements

The entire team participates in a project kickoff meeting where Ted and Vince present the business objectives of the project. The business requirement is to improve promotion effectiveness based on high-end television products. Specifically:

- Characterize distinct shopping behavioral segments for customers who have previously purchased televisions.
- For each of these customer segments, discover affinities among televisions and other items in other departments.
- Identify the next most likely purchase for each customer segment.
- Increase the average spend of customers in the most profitable clusters.
- Increase the average spend of the less profitable customers who look like the more profitable customers (that is, in the same clusters) by cross-selling or up-selling them the products that the more profitable customers have bought.

Related to the above objectives, there are specific business questions about which Ted and Vince would like to have more insight:

- What is the shopping behavior for TV customers?
- What do the TV customers look like?
- What products are sold in the same transaction with televisions?
- What products will a TV customer buy next?
- What customers should be targeted in a campaign?

The project team evaluates all the business requirements and compiles a list of components required to build the solution, including Web applications, data mining models, and other technical resources. Wendy documents all the major points of the discussion and includes them in a detailed project plan.

9.3.5 The data mining approach

Olivia evaluates the business requirements and decides that the best data mining approach to support this project is to use the data mining discovery methods along with scoring for real-time cross-sell and up-sell opportunities.

One of the business objectives is to boost storewide sales based on high-end TV shoppers. A way to accomplish this is to understand the behavior of customers who had purchased TV in the past to determine their shopping preferences. A data mining customer segmentation model would be ideal to discover the shopping behavior of customers based in their past transactions.

Additionally, a data mining segmentation model would also provide a way to discover segments of highly profitable customers to be targeted in future promotions. Value Trend has a large number of customers shopping in their stores, but just a small percentage shop in the electronics departments. The ability to identify new customers who behave like TV customers is essential for target promotions. Behavioral attributes combined with some demographics would be appropriate to understand customer behavior and find profitable segments.

Value Trend also would like to increase wallet share for internet and store shoppers. That means getting customers to add more items to the basket whenever they perform a transaction (store or internet). A data mining associations model provides rules to discover items that tend to be sold in the same transaction.

Since the future promotions are targeted to electronics (high-end TV) shoppers, a data mining associations model using sales transactions of customers who had purchased a TV in the past 12 months would provide a good insight to identify what additional items those customers usually purchase along with TVs. And, specifically, what products are sold in the same transaction with plasma and LCD TVs.

Value Trend management also would like to anticipate future sales by understanding what products a TV customer is more likely to purchase next. Answering this question would provide a great opportunity to make special offerings to customers purchasing high-end TVs and anticipate revenue with next-most-likely items that those customers would purchase. That would not only anticipate the revenue but also keep those customers from shopping with Value Trend's competitors. A data mining sequences model provides rules to determine patterns of items that are purchased in sequences.

Olivia also suggests the use of a real-time scoring application so that new customers shopping through the internet could be scored and have a instant offering based on their purchase preferences. A similar application could be used at the cash register in the stores, and instant discounts coupons could be used to induce customers to return to the stores or shop in different departments during the same visit.

9.3.6 The data preparation process

This step describes the process used to obtain and prepare the data for the data mining models.

Data selection

Simon identifies four major tables to used in the process. These tables are as depicted in Table 9-1.

Table 9-1 Tables for data selection

Table name	Table description
IDV	Individuals attributes, such as gender, marital status, income range, and others.
MSR_PRD	Time Periods, such as calendar and fiscal years, quarters, months, day of week, and others.
PD	Product information, such as product department, class, sub class, and others.
PRCHS_PRFL_ANALYSIS	Customer profiles analysis. This tables contain records of customers purchase transactions.

Figure 9-6 shows the layout for the PRCHS_PRFL_ANLYSIS table. This table contains detailed transaction record information. There is one record for every transaction a given customer performed on a given day at a given store. This table is the basis to provide the information for clustering and associations. In clustering models, this data requires additional transformations and aggregations. Associations models use the information at the transactional level. This table contains three years of transactional sales information, but for the scope of the analysis in this data mining project only transactions from the last 12 months are selected.

Layout for Table PRCHS_PRFL_ANLYSIS

Column Name	Data Type	Definition
IDV_IP_ID	INTEGER	Foreign Key for Individuals (IDV) table
OU_IP_ID	INTEGER	Foreign Key for Organization Unit (Store) table.
PD_ID	INTEGER	Foreign Key for Products (PD) table
MSR_PRD_ID	SMALLINT	Foreign Key for Time Periods (MSR_PRD) table
MKT_BSKT_TXN_ID	INTEGER	Foreign Key for Market Baskets table
NMBR_OF_MRKT_BSKTS	INTEGER	Number of Market Baskets
NUMBER_OF_ITEMS	INTEGER	Number of items purchased in a given transaction
PRDCT_BK_PRC_AMUNT	DECIMAL	Product Price Amount
CST_OF_GDS_SLD_CGS	DECIMAL	Cost of Goods Sold for a given transaction
SALES_AMOUNT	DECIMAL	Total Sales Amount for a given transactions

Figure 9-6 Customer purchase profiles: transactional record detail

Olivia and Simon identify and select the basic product and demographic information to be used in the data mining model. Figure 9-7 shows the selected list of attributes (columns) from the source tables PD (Products) and IDV (Individuals). Some of the product attributes are used for product taxonomy in the associations and sequences models. The individual attributes are selected for use in the clustering models.

Attributes from Source PD (Products) Table		
Column Name	**Data Type**	**Definition**
PD_ID	INTEGER	Primary Key for Product (PD) table
NM	VARCHAR	Product Name
PD_DEPT_NM	VARCHAR	Product Department Name (For example, Electronics and Computers)
PD_SUB_DEPT_NM	VARCHAR	Product Sub Department Name (For example, Televisions)
PD_CL_NM	VARCHAR	Product Class Name (For example, Plasma TV)
...		

Attributes from Source IDV (Individuals) Table		
Column Name	**Data Type**	**Definition**
IDV_IP_ID	INTEGER	Primary Key for Individuals (IDV) table
GND_NM	VARCHAR	Gender Name (For example, Male and Female).
IDV_AGE_RNG_NM	VARCHAR	Individual Age Range (For example, 20-25 years)
IDV_MAR_ST_TP_NM	VARCHAR	Individual Marital Status (For example, Married or Single)
IDV_GEODEMOGGRP_NM	VARCHAR	Individual Demographic Group (For example, Low Income Families or Affluent Families)
IDV_INCM_RNG_NM	VARCHAR	Individual Income Range (For example, 60k–100k) per year
...		

Figure 9-7 PD (Products) and IDV (Individuals) table layout for selected attributes

Defining the analytical variables

Olivia and Simon identify the need to create additional analytical variables to be used in the data mining clustering model. Figure 9-8 shows a list of variables to be defined. Most of the behavioral variables are derived from the sales amount field of the PRCHS_PRFL_ANLYSIS table. The demographic variables already exist in the customer table (IDV table).

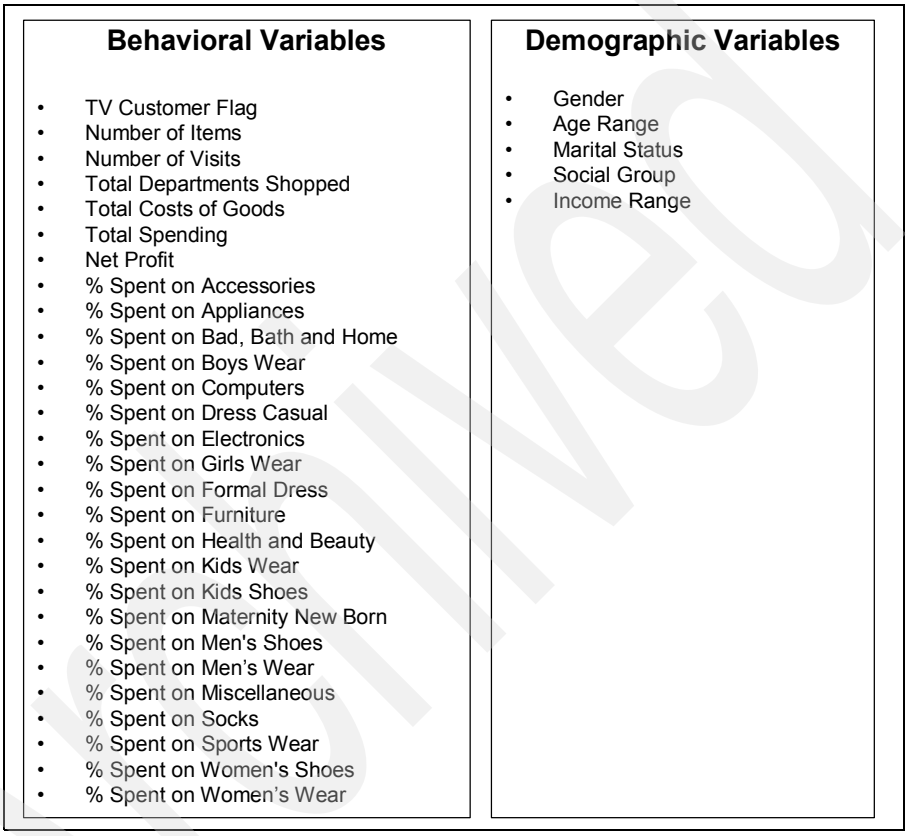


Figure 9-8 Analytic variables

The TV Customer Flag variable provides more clarification in the analysis to easily identify the TV customers segments. The % Spent variables show how customers allocate their spending across departments.

Defining the target table layouts

Based on the data mining approach defined for the project (clustering and associations), specific data layouts are required to support the data mining models. The data layouts to support the data mining models are the behavioral-demographic and transactional layouts. Details on these two types of data layouts are defined in Chapter 6, “Data preparation for data mining” on page 97.

Figure 9-9 shows the layout for the behavioral-demographic table for the data mining clustering model. Notice that all variables representing the customer spending behavior are defined in percentages instead of using the dollar sales amount.

Customer Purchase Behavior and Demographic Table Layout		
Column Name	**Type**	**Definition**
CUSTOMER_ID	Integer	Customer Unique Identification Number
TV_CUSTOMER	Char	Y/N Flag to define if the customer had purchase TV items in the past 12 months
NUM_ITEMS	Integer	Number of Items customer have purchased in the past 12 months
NUM_VISITS	Integer	Number of Visits the customer had performed (store visits and internet visits) in the past 12 months
TOTAL_DEPT_SHOPPED	Integer	Total Departments shopped in the past 12 months
TOTAL_COGS	Decimal	Total cost for all merchandise purchased in the past 12 months
TOTAL_SPENDING	Decimal	Total customer spending in the past 12 months
NET_PROFIT	Decimal	Net Profit for the past 12 months
PCT_SPNT_ACCESSORIES	Decimal	Percent spending on accessories department in the past 12 months.
PCT_SPNT_APPLIANCES	Decimal	Percent spending on appliances department in the past 12 months.
PCT_SPNT_COMPUTERS	Decimal	Percent spending on computers department in the past 12 months.
PCT_SPNT_ELECTRONICS	Decimal	Percent spending on electronics department in the past 12 months.
...	Decimal	Percent spending on … department in the past 12 months.
...	Decimal	
PCT_SPNT_SPORTS_WEAR	Decimal	Percent spending on sports wear department in the past 12 months.
PCT_SPNT_WOMENS_SHOES	Decimal	Percent spending on women's shoes department in the past 12 months.
PCT_SPNT_WOMENS_WEAR	Decimal	Percent spending on women’s wear department in the past 12 months.
GENDER	Char	Customer gender
AGE_RANGE	Char	Customer age range
MARITAL_STATUS	Char	Customer marital status
SOCIAL_GROUP	Char	Customer social group
INCOME_RANGE	Char	Customer income range

Figure 9-9 Customer purchase behavior and demographic table layout

Figure 9-10 shows the layout for the transactional table defined for data mining associations and sequences models. The associations model requires only two of the variables (ITEM_ID and TRANSACTION_DATE).

Customer Transactions Table Layout		
Column Name	**Type**	**Definition**
CUSTOMER_ID	Integer	Customer Unique Identification Number
ITEM_ID	Integer	Product Identification Number
TRANSACTION_DATE	Date	Customer Purchase Transaction Date

Figure 9-10 Customer transactions table layout

Defining the ETL process

After gathering all components (identified source data, data mining variables, and target tables) for the data preparation process, the next step is to perform the ETL process. Simon documents all transformations required to generate the data for the data mining process. Since Value Trend acquired DB2 Data Warehouse software from IBM, Simon and Wendy decides it would be a good opportunity to leverage DB2 Data Warehouse SQL Warehousing tooling to help in this activity. Figure 9-11 shows some of the calculations required to generate the customer behavior and demographic table.

```
TOTAL_ DEPT_SHOPPED = COUNT(DISTINCT DEPT_NAME) GROUP BY CUSTOMER_ID

NUM_VISITS = COUNT(DISTINCT TRANSACTION DATE) GROUP BY CUSTOMER_ID

NUM_ITEMS = COUNT(DISTINCT ITEM_ID GROUP BY CUSTOMER_ID

TOTAL_SPENDING = SUM (SALES AMOUNT) GROUP BY CUSTOMER_ID

PCT_SPNT_ELECTRONICS = (TOTAL SPENT ELECTRONICS / TOTAL SPENT AMOUNT) * 100 GROUP BY
CUSTOMER_ID
```

Figure 9-11 Sample calculations

Using DB2 Warehouse Design Studio, Simon creates a data flow to populate the target customer behavior demographic table. The data flow is represented in Figure 9-12. For easy of understanding, a number is assigned to each operator and described below.

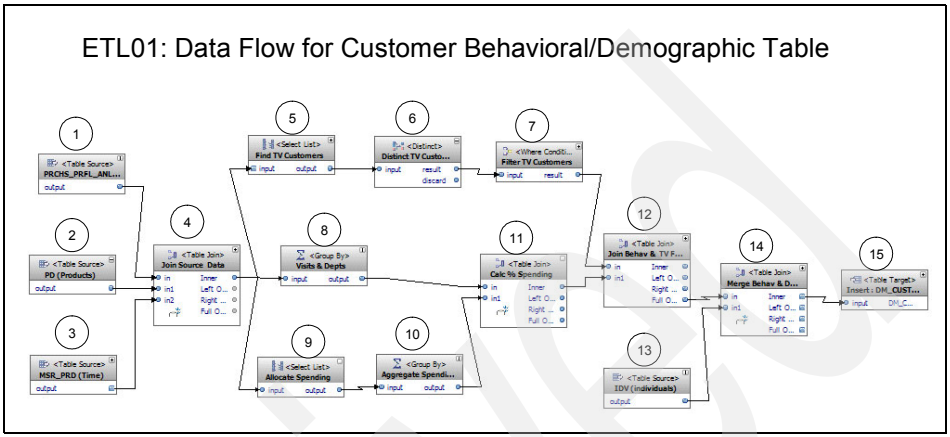


Figure 9-12 Data flow for customer behavioral-demographic table

Operators 1, 2, and 3: Table Source operators to select data from tables PD (product), MSR_PRD (Time), and PRCHS_PRFL_ANLYSIS (Customer Transactions details). Also filters the selection of transactions for the last 12 months.

Operator 4: Table Join operator to join tables: PD (product), MSR_PRD (Time), and PRCHS_PRFL_ANLYSIS (Customer Transactions details).

Operators 5, 6 and 7: Creates a flag (Y/N) for all customers that had purchased at least one TV item in the past 12 months.

Operator 8: Creates variables NUM_VISITS and TOTAL_DEPT_SHOPPED.

Operator 9: Allocates customer spending by department.

Operator 10: Calculates total customer spending by department.

Operator 11: Calculates customer spending percentages.

Operator 12: Merges behavioral data with the TV flag.

Operator 13: Selects demographic information from the IDV (Individuals) table.

Operator 14: Merges behavioral and demographic data.

Operator 15: Inserts behavioral and demographic data into the target table (DM_CUST_BEHAV_DEMOG) Table.

Figure 9-13 shows sample data for a few variables from the customer behavior-demographic table.

Customer Behavioral/Demographic Table – Sample Data

Cust ID	TV Shopper	# Items	# Visits	# Depts Shopped	Tot $ Spending	% Spent ...	% Spent Computers	% Spent Electronics	Gender	Age Range
1635	Y	8	1	3	1,234.87	...	87.78	6.83	Female	55 - 64
1890	Y	8	2	3	1,371.45	...	85.85	5.50	Female	45 - 54
631	Y	14	4	3	2,548.39	...	85.72	13.41	Male	55 - 64
2305	Y	9	2	3	2,010.37	...	73.30	20.80	Female	45 - 54
1772	Y	10	4	3	1,684.06	...	66.89	31.16	Female	45 - 54
1873	Y	3	2	2	68.58	...	66.66	33.33	Male	45 - 54
704	Y	14	3	2	7,481.78	...	59.88	40.11	Male	55 - 64
1035	Y	8	2	2	2,462.54	...	43.33	56.66	Female	55 - 64
671	Y	8	3	3	3,342.56	...	42.57	57.33	Male	65+
2000	Y	12	3	2	987.33	...	39.32	60.67	Female	45 - 54
2236	Y	7	2	3	104.79	...	36.36	37.01	Male	45 - 54
2076	Y	6	4	4	117.20	...	27.45	29.46	Male	45 - 54
2596	Y	9	3	3	4,733.42	...	26.48	73.19	Female	35 - 44
1881	Y	6	1	3	544.04	...	14.00	82.50	Male	45 - 54
2071	Y	3	2	2	153.72	...	4.30	95.69	Male	45 - 54
1931	Y	4	2	2	2,203.36	...	4.22	95.77	Female	45 - 54
513	Y	4	2	2	1,454.41	...	3.95	96.04	Female	65+
2449	Y	7	4	3	503.34	...	3.32	92.32	Female	45 - 54
817	Y	6	3	2	3,747.68	...	3.16	96.83	Male	55 - 64
2257	Y	109	65	5	11,694.20	...	2.89	91.02	Male	45 - 54
1510	Y	4	1	2	2,794.71	...	2.41	97.58	Female	45 - 54
1993	Y	6	3	3	1,591.95	...	1.64	97.08	Female	45 - 54
154	Y	9	4	3	3,208.20	...	0.89	98.32	Female	65+
2419	Y	10	3	3	2,162.98	...	0.25	98.91	Male	45 - 54
808	Y	5	2	3	1,809.70	...	0.18	99.63	Male	55 - 64
24	N	2	2	2	24.65	...	0.00	0.00	Male	65+
25	N	2	1	1	25.42	...	0.00	0.00	Male	65+
26	N	1	1	1	30.33	...	0.00	0.00	Female	65+
27	N	4	1	1	12.50	...	0.00	0.00	Female	65+

Commit | Roll Back | Fetch More Rows

Figure 9-13 Customer behavioral-demographic table: sample data

9.3.7 The data mining process

Olivia uses DB2 Warehouse to create mining flows for data mining clustering and scoring models. Olivia creates and executes the data mining flows with different settings until she achieves the desired results. Figure 9-14 shows the mining flow that includes the clusterer and scoring operators. She uses the Kohonen neural clustering algorithm because it provided better results compared to demographic clustering in this case.

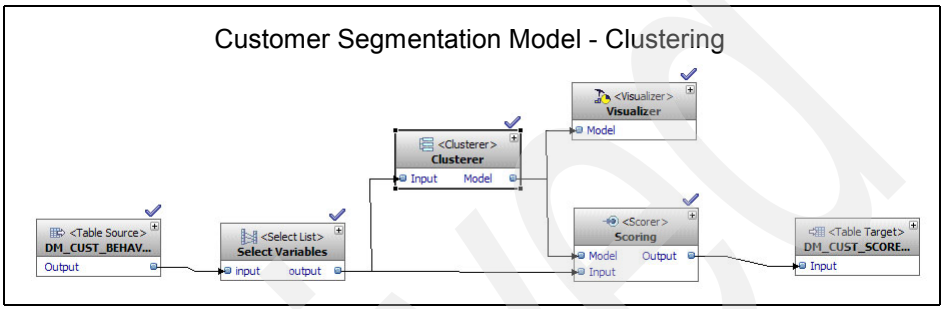


Figure 9-14 Design Studio mining flow for customer segmentation

After identifying the target segments in the clustering model, Olivia created an associations model to discover product affinities rules among transactions performed by customers in the target segments. Identifying product affinity rules would not only be beneficial for the marketing group for target promotions, it also would provide insights for buyers and merchandise planners to anticipate product inventories required to support the promotion. Figure 9-15 shows the data mining flow used for the associations model. The join is used to select only the purchase transactions from existing TV customers.

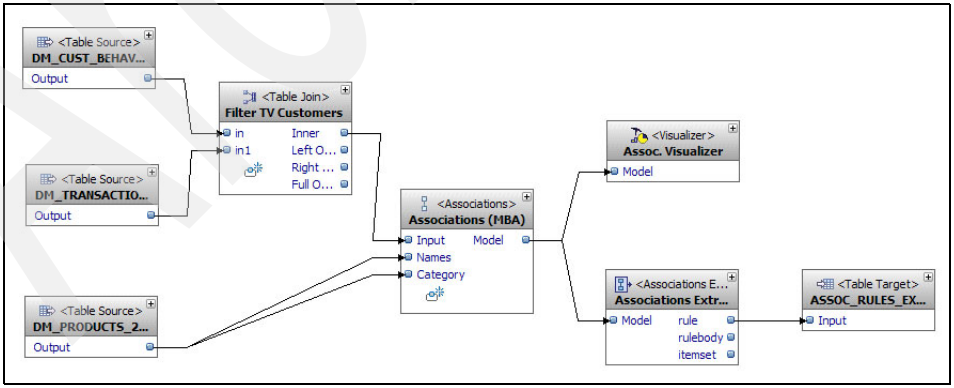


Figure 9-15 Market basket analysis (MBA): Associations Model

9.3.8 Analyzing results

Olivia performs several iterations to refine the data mining models. After achieving a good model quality and reasonable results, she involves other members of the team to validate the data mining findings.

Clustering analysis

The customer segmentation model included all customers (TV Shoppers and Non-TV Shoppers). Figure 9-16 shows the clustering results. Nine distinct clusters were identified. The clusters are ordered by size, and the variables in the cluster are ordered by order of importance in distinguishing the members of the cluster from all other customers.

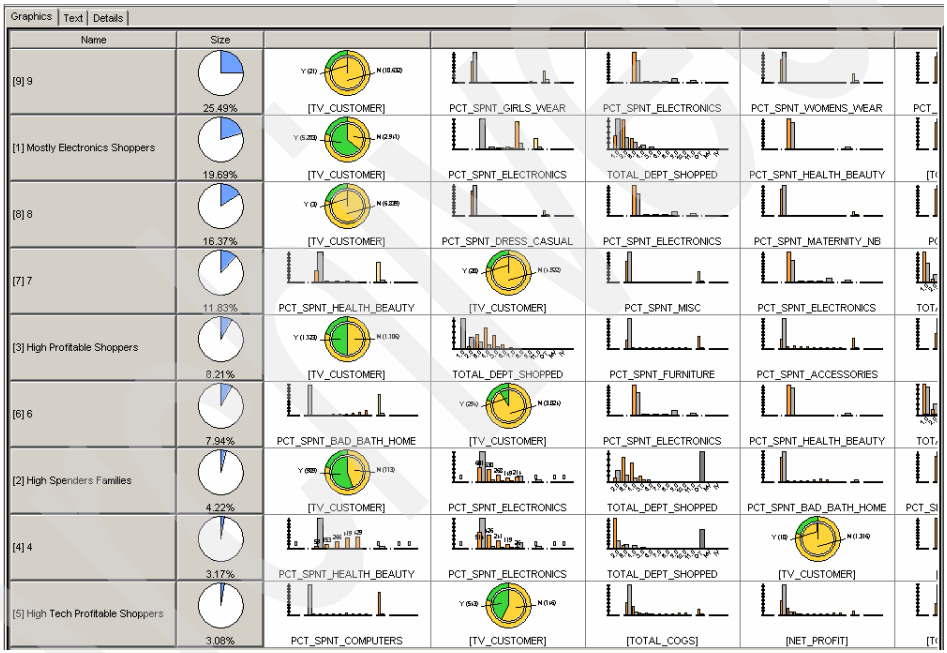


Figure 9-16 Clustering visualization results

Olivia analyzes the data mining segmentation model and identified four clusters (clusters #1, #3, #2, and #5) that are characterized as TV shoppers. She names the clusters as shown in the Figure 9-17.

Cluster #1 contains 8,230 customers and represents 19.69% of the overall population. Cluster #3 contains 3,429 customers and represents 8.21% of the overall population. Cluster #2 contains 1,762 customers and represents 4.22% of the overall population. Cluster #5 contains 1,289 customers and represents 3.08% of the overall population. The four clusters together contain 14,710 customers, and they represent 35.20% of the overall population.

Cluster#/ Name	Cluster Characteristics
[1] Mostly Electronics Shoppers Size = 19.69%	[TV_CUSTOMER] happens to be predominantly **Y**, PCT_SPNT_ELECTRONICS is **high**, TOTAL_DEPT_SHOPPED is predominantly 2.0, PCT_SPNT_HEALTH_BEAUTY is medium, [TOTAL_SPENDING] happens to be medium, [TOTAL_COGS] happens to be medium, PCT_SPNT_GIRLS_WEAR is medium, PCT_SPNT_WOMENS_WEAR is medium, PCT_SPNT_BAD_BATH_HOME is medium, NUM_ITEMS is medium, PCT_SPNT_MISC is medium, [NET_PROFIT] happens to be medium, NUM_VISITS is medium, PCT_SPNT_BOYS_WEAR is medium, and PCT_SPNT_KIDS_WEAR is medium.
[3] High Profitable Shoppers Size = 8.21%	[TV_CUSTOMER] happens to be predominantly **Y**, TOTAL_DEPT_SHOPPED is predominantly 4.0, PCT_SPNT_FURNITURE is **high**, PCT_SPNT_ACCESSORIES is **high**, NUM_ITEMS is medium, [TOTAL_COGS] happens to be high, [NET_PROFIT] happens to be **high**, [TOTAL_SPENDING] happens to be **high**, NUM_VISITS is **high**, PCT_SPNT_ELECTRONICS is medium, PCT_SPNT_APPLIANCES is **high**, PCT_SPNT_COMPUTERS is medium, PCT_SPNT_HEALTH_BEAUTY is medium, PCT_SPNT_GIRLS_WEAR is medium, and PCT_SPNT_BAD_BATH_HOME is medium.
[2] High Spenders Families Size = 4.22%	[TV_CUSTOMER] happens to be predominantly **Y**, PCT_SPNT_ELECTRONICS is medium, TOTAL_DEPT_SHOPPED is predominantly 3.0, PCT_SPNT_BAD_BATH_HOME is medium, PCT_SPNT_WOMENS_WEAR is medium, NUM_VISITS is high, NUM_ITEMS is medium, [TOTAL_COGS] happens to be **high**, [NET_PROFIT] happens to be **high**, [TOTAL_SPENDING] happens to be **high**, PCT_SPNT_GIRLS_WEAR is medium, PCT_SPNT_APPLIANCES is medium, PCT_SPNT_HEALTH_BEAUTY is medium, PCT_SPNT_KIDS_WEAR is medium, and PCT_SPNT_BOYS_WEAR is medium.
[5] High Tech Profitable Shoppers Size = 3.08%	PCT_SPNT_COMPUTERS is predominantly **90** - +8 , [TV_CUSTOMER] happens to be predominantly N, [TOTAL_COGS] happens to be high, [NET_PROFIT] happens to be **high**, [TOTAL_SPENDING] happens to be **high**, TOTAL_DEPT_SHOPPED is predominantly 4.0, PCT_SPNT_ELECTRONICS is medium, PCT_SPNT_FURNITURE is medium, NUM_ITEMS is medium, NUM_VISITS is **high**, PCT_SPNT_APPLIANCES is **high**, PCT_SPNT_HEALTH_BEAUTY is medium, PCT_SPNT_GIRLS_WEAR is medium, PCT_SPNT_WOMENS_WEAR is medium, and PCT_SPNT_BAD_BATH_HOME is medium.

Figure 9-17 TV Customer cluster analysis

Figure 9-18 shows the details for all clusters. Overall cluster quality is 0.864, indicating a good quality model.

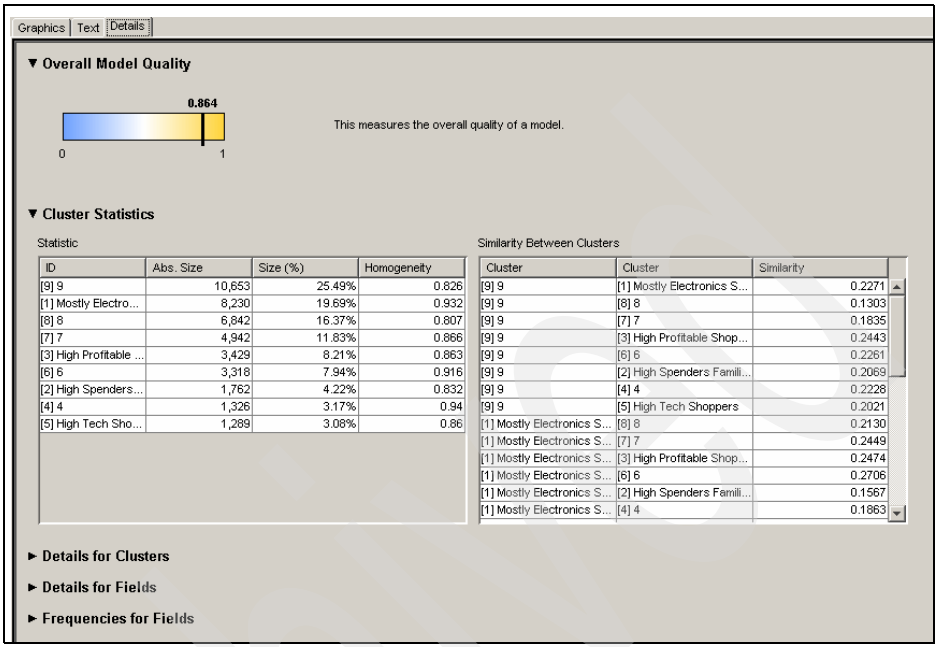


Figure 9-18 Clustering details

Figure 9-19 shows the variables for cluster #1. The three most important variables that characterize this cluster are, by order of importance: TV_CUSTOMER, PCT_SPNT_ELECTRONICS, and TOTAL_DEPT_SHOPPED.

There are 5,283 TV customers in these clusters. Customers in this cluster tend to concentrate their purchases in two departments, and most of them have most (90%-100%) of their spending in the electronics department.

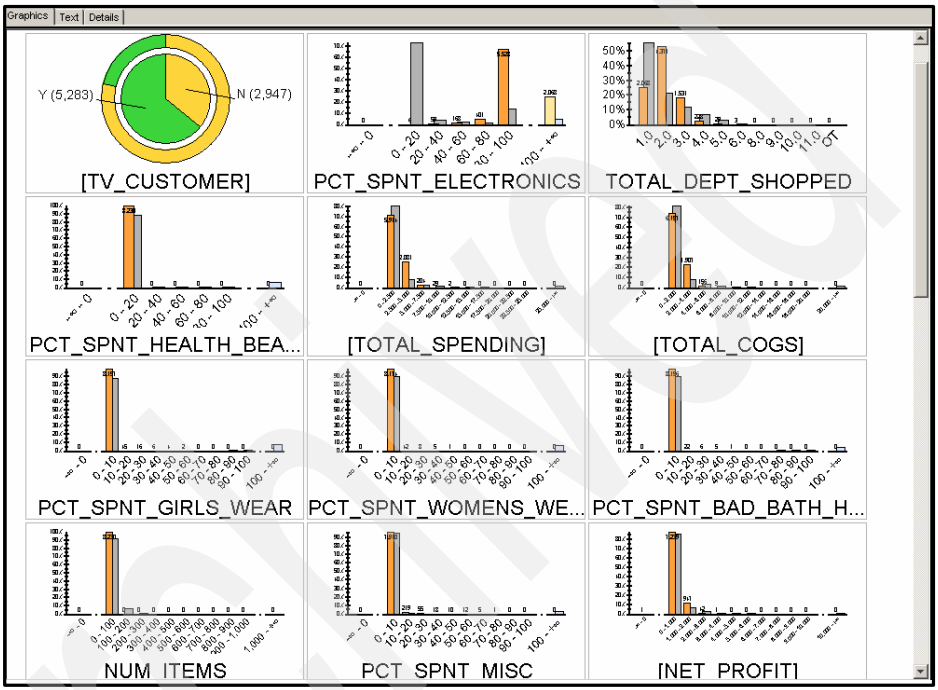


Figure 9-19 Cluster #1: "Mostly Electronics Shoppers"

Figure 9-20 shows the distribution of TV Customers (Y) versus non-TV Customers (N) in cluster #1 compared to the overall population. It is important to note that this cluster contains 5,283 customers who already purchased a TV in the past 12 months, and there are 2,497 customers who have not purchased a TV yet but have spending behavior in other departments very similar to customers who have purchased a TV. Those are very good candidates to be targeted in a TV promotion.

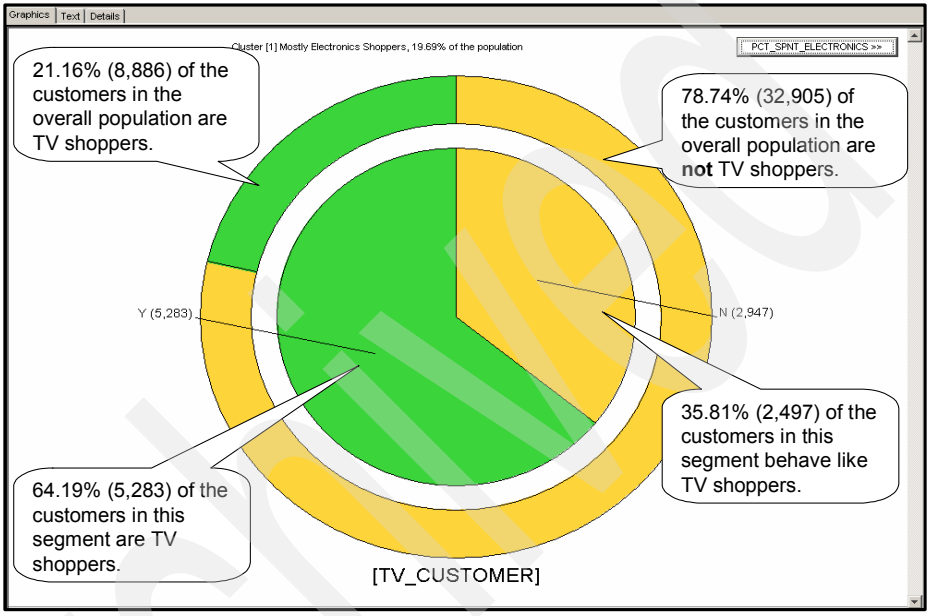


Figure 9-20 Cluster #1: TV customer Indicator variable

Figure 9-21 shows the customers spending in the electronics department. Customers in cluster #1 spend more in electronics than the overall population. For example, 67.17% of customers in cluster #1 spend between 80% and 100% of their total spending in the electronics department; however, in the overall population, only 13.8% of the customers have similar spending behavior.

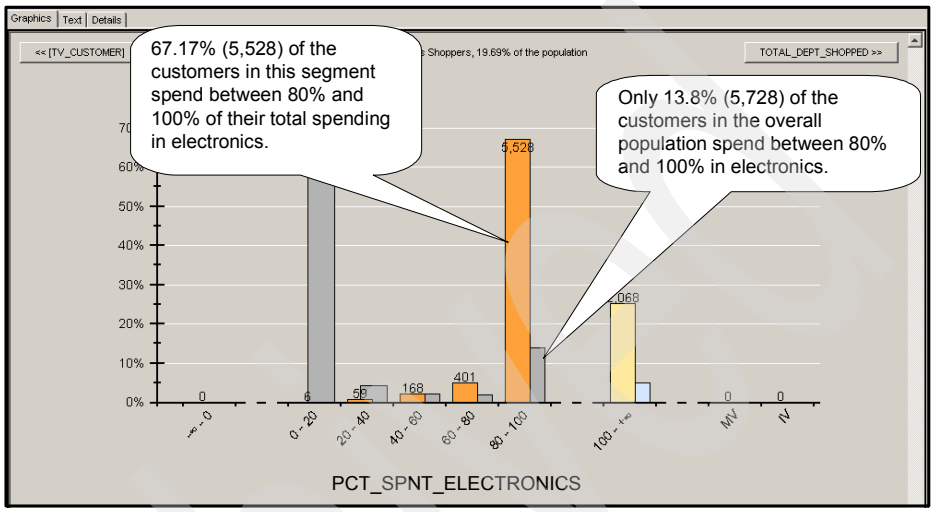


Figure 9-21 Cluster #1: Customer spending in electronics

Cluster #3 (Figure 9-22) consists of profitable customers who shop in multiple departments. They have above-average spending in furniture and accessories. This is a good segment to target in a TV promotion. There are 1,706 customers in this segment who did not purchase TV items, but they have similar shopping behavior as the customers in this segment who did purchase TVs. Further clustering analysis could be performed with just the members of this segment to further understand the distinct characteristics of these customers.

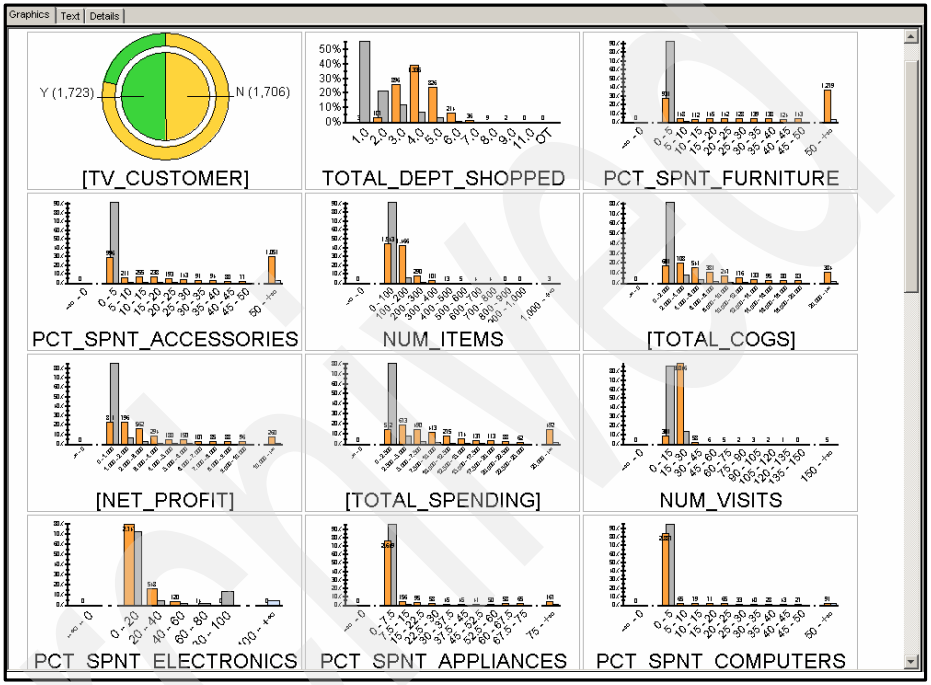


Figure 9-22 Cluster #3: "High Profitable Shoppers"

Cluster #2 (Figure 9-23) consists of profitable customers who shop in multiple departments. They are high spenders in electronics, women's wear, bed-bath, and home products. This is also a good segment to be targeted in a TV promotion. There are 773 customers in this segment who did not purchase TV items, but they have shopping behavior similar to the customers in this segment who have purchased TVs.

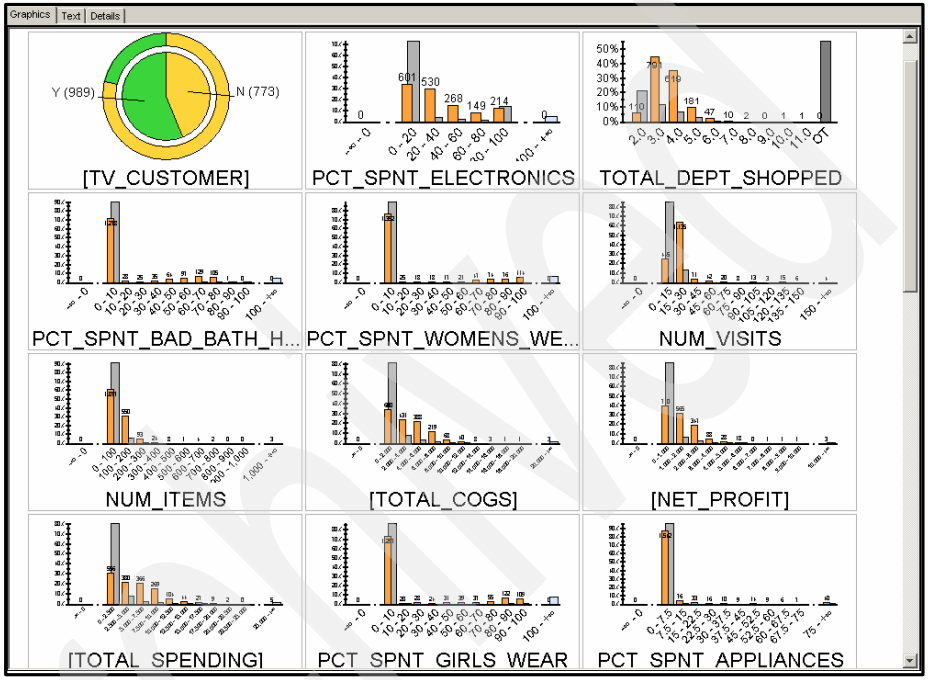


Figure 9-23 Cluster #2: "High Spenders Families Shoppers"

Cluster #5 (Figure 9-24) consists of profitable customers who shop in multiple departments. They are high spenders in computers, electronics, and furniture. This is also a good segment to be targeted in a TV promotion. There are 746 customers in this segment who did not purchase TV items, but they have shopping behavior similar to the customers in this segment who did purchase TVs.

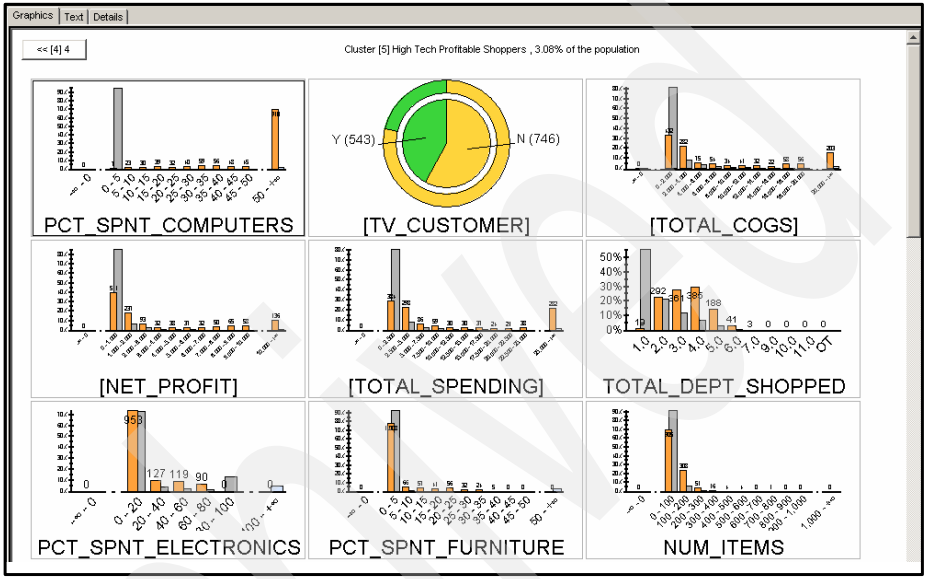


Figure 9-24 Cluster #5: "High Tech Profitable Shoppers"

Associations analysis (MBA)

Once the customer segmentation model is completed and target clusters are identified, the next step in the overall data mining process is to find item affinities (associations rules) based on the transactions of customers in the target clusters.

Figure 9-25 shows the results for the associations model. The *rules* column shows the affinities rules, where [body item(s)] => [head item]. The *support* column shows the percentage of all transactions that include the items in the rule. The *confidence* column shows the percentage of transactions containing the body item(s) that also contain the head item. The *lift* column shows lift, which indicates the strength of the rule. A lift of 1 means that the body item has no effect on the presence of the head item in a transaction. A lift less than 1 means that the body and head items are substitutes for one another. A lift greater than 1 means that head item is more likely to be present when the body item is also present. The *absolute support* column is the number of transactions containing the rule.

Market Basket Analysis (MBA) - Associations Results

Rules | Item Sets | Graph | Statistics

Visible rules:

Rule	Support	Confidence	▼ Lift	Absolute Support	Subtractive Lift	Items in Rule Body	Items in Rule Head
[PROJECTION TV] ==> [COMBO TV]	19.8347%	50.3497%	1.26	144	0.10	1	
[PROJECTION TV] ==> [CLEANING CARE SETS]	30.3030%	76.9231%	1.22	220	0.14	1	
[PROJECTION TV] ==> [GAMING CONSOLE]	29.2011%	74.1259%	1.21	212	0.13	1	
[COMBO TV] ==> [GAMING CONSOLE]	28.9256%	72.1649%	1.18	210	0.11	1	
[PROJECTION TV] ==> [LCD FLAT PANEL]	39.1185%	99.3007%	1.14	284	0.12	1	
[COMBO TV] ==> [LCD FLAT PANEL]	39.2562%	97.9381%	1.13	285	0.11	1	
[COMBO TV] ==> [CLEANING CARE SETS]	28.3747%	70.7904%	1.12	206	0.08	1	
[PROJECTION TV] ==> [PLASMA FLAT PANEL]	38.9807%	98.9510%	1.12	283	0.10	1	
[LCD FLAT PANEL] ==> [CLEANING CARE SETS]	61.0193%	70.2060%	1.12	443	0.07	1	
[PLASMA FLAT PANEL] ==> [CLEANING CARE SETS]	61.9835%	69.9844%	1.11	450	0.07	1	
[COMBO TV] ==> [PLASMA FLAT PANEL]	39.3939%	98.2818%	1.11	286	0.10	1	
[COMBO TV] ==> [CELLULAR PHONES]	36.0882%	90.0344%	1.11	262	0.09	1	
[PLASMA FLAT PANEL] ==> [LCD FLAT PANEL]	85.2617%	96.2675%	1.11	619	0.09	1	
[LCD FLAT PANEL] ==> [PLASMA FLAT PANEL]	85.2617%	98.0963%	1.11	619	0.10	1	
[LCD FLAT PANEL] ==> [GAMING CONSOLE]	58.4022%	67.1949%	1.10	424	0.06	1	
[PLASMA FLAT PANEL] ==> [GAMING CONSOLE]	59.0909%	66.7185%	1.09	429	0.06	1	
[PROJECTION TV] ==> [CELLULAR PHONES]	34.8485%	88.4615%	1.09	253	0.07	1	
[LCD FLAT PANEL] ==> [CELLULAR PHONES]	75.4821%	86.8463%	1.07	548	0.06	1	
[PLASMA FLAT PANEL] ==> [CELLULAR PHONES]	76.5840%	86.4697%	1.06	556	0.05	1	
[COMBO TV] ==> [ACCESSORIES AUDIO]	40.0826%	100.0000%	1.06	291	0.06	1	
[PROJECTION TV] ==> [ACCESSORIES AUDIO]	39.3939%	100.0000%	1.06	286	0.06	1	
[PLASMA FLAT PANEL] ==> [ACCESSORIES AUDIO]	88.4298%	99.8445%	1.06	642	0.06	1	
[LCD FLAT PANEL] ==> [ACCESSORIES AUDIO]	86.7769%	99.8415%	1.06	630	0.06	1	
[COMBO TV] ==> [VIDEO CAMERAS]	37.7410%	94.1581%	1.03	274	0.03	1	
[PROJECTION TV] ==> [VIDEO CAMERAS]	36.7769%	93.3566%	1.02	267	0.02	1	
[LCD FLAT PANEL] ==> [VIDEO CAMERAS]	80.9917%	93.1854%	1.02	588	0.02	1	
[PLASMA FLAT PANEL] ==> [VIDEO CAMERAS]	82.2314%	92.8460%	1.02	597	0.02	1	
[PLASMA FLAT PANEL] ==> [DVD PLAYERS]	87.4656%	98.7558%	1.01	635	0.01	1	
[LCD FLAT PANEL] ==> [DVD PLAYERS]	85.8127%	98.7322%	1.01	623	0.01	1	
[PROJECTION TV] ==> [DVD PLAYERS]	38.8430%	98.6014%	1.01	282	0.01	1	
[COMBO TV] ==> [DVD PLAYERS]	39.3939%	98.2818%	1.01	286	0.01	1	

Figure 9-25 Associations rules for TV shoppers transactions

Olivia observes strong affinities between televisions and several other items in the electronics department. For example, the third rule [PROJECTION TV] ==> [GAMING CONSOLE] has a support of 29.2%, indicating that the Projection TV and Gaming Console are purchased together in 29.2% of all the transactions by TV shoppers. This rule has a confidence of 79.1%, indicating that 79.1% of the transactions that contain a Projection TV also contain a Gaming Console. Cleaning Care Sets, Cellular Phones, Audio Accessories, and Video Cameras

are other examples of items that have strong affinities with TV items. These rules should be considered in anticipation of a promotion for TV items. For example, if a marketing promotion of Projection TV items expects to increase sales of Projection TVs by 10%, this would yield a revenue increase from Gaming Consoles of 7.9% (equal to 10% x 79.1%). For a successful promotion, not only TV Items should be stocked, but also the items with affinities rules to the TV items.

9.3.9 Business actions

The VP of marketing selects four high-potential segments of customers. The selected segments represented 6,172 customers who have similar behavior to customers that shop television products. Some of these customers shop across multiple departments, and have high spending levels. The four segments are summarized in Figure 9-26.

Cluster Name	Number of Target Customers in Cluster	Expected Number of Responders to Promotion @6% Response Rate	Expected Revenue for TV Items @ Average Price of $1,300 per unit	Expected Revenue for Gaming Console @ $300 per unit	Expected Total Revenue
Mostly Electronics Shoppers	2,947	177	$229,866	$39,338	$269,204
High Profitable Shoppers	773	46	$60,294	$10,318	$70,612
High Profitable Families Shoppers	1,706	102	$133,068	$22,773	$155,841
High Tech Profitable Shoppers	746	45	$58,188	$9,958	$68,146
Total	6,172	370	$481,416	$82,388	$563,804

Figure 9-26 Expected revenue from a targeted promotion for TV items

The plan is to design targeted promotional offers to appeal to the distinct characteristics of each target segment as revealed by the clustering model. Targeted promotional offers are based on associations relationships. Assumptions are a 6 percent response rate to the offers. Given the average price of $1,700 for high-end television items, the expected revenue for televisions is $481,416 in response to a promotion. Additional revenue is also expected from the associated items.

As a simplistic example to illustrate how total revenue from the promotion was calculated, consider the values in Figure 9-26 on page 279 and the third rule (highlighted) in Figure 9-25 on page 278. This single rule indicates that the affinity between the Projection TV and the Gaming Console would yield expected incremental revenue as follows. In Figure 9-25 on page 278, the Gaming Console is present in 74.1% (confidence) of the transactions that include the Projection TV. Given a price of $300 for a Gaming Console, the expected incremental revenue is $82,388, calculated using the following formula:

370 Responders x 0.741 x $300 = $82,388

Of course, the 370 responders would probably be purchasing a variety of high-end televisions, not just the Projection TV. But this simple example illustrates how incremental expected revenue from affinity items can be calculated based on associations rules.

Considering the expected revenue from all high-end televisions and their associated items, the marketing team justifies the promotion based on ROI. The promotion design includes stocking requirements based on expected increases in sales of televisions and related items as indicated by the data mining results from associations.

10

Emerging applications of data mining

In this chapter, we sketch some recent trends and applications of data mining. Common to all of them is that they broaden the scope of data mining. They do this either by combining data mining with other analysis techniques such as text analysis or OLAP, by providing an easier and more flexible access to data mining, such as through Web services, or by applying data mining in new areas and use cases. For each of these three aspects of broadening the scope, this chapter contains one or two examples.

10.1 Incorporating unstructured text into data mining

Data mining techniques are traditionally applied to structured data, which most frequently resides in relational data bases. However, most experts agree that between 70% and 90% of the data that is stored in a typical enterprise is unstructured data, such as letters, e-mails, notes, plans, scenarios, and so on. One way of making this vast reservoir of interesting information accessible for data mining techniques is to use text annotation and text analysis techniques that enrich raw texts with semantic tags, extracted keywords, concepts, or sentiments.

Here we describe, by means of a concrete sample use case, how the results of a text analysis on unstructured textual information can be used together with structured data in data mining. More precisely, we show how to combine the two types of data sources for:

- Detecting typical rules and patterns in the data
- Creating a prediction model that is able to predict future events based on knowledge learned from historic data

Our sample scenario is the evaluation of a medical study on the mortality risk of patients with coronary cardiac diseases. During the study, both structured information such as age, blood pressure, and cholesterol values as well as unstructured information such as medical histories have been collected for a certain number of patients.

Other typical use cases where the combination of structured and unstructured data sources is highly beneficial are:

- Customer relationship management based on demographic customer data (structured), transaction data (structured), and e-mails, call center logs, or customer agent notes (unstructured).
- Causal analysis of errors, failures, breakdowns or other incidents, based on structured information about involved components, processes or times, and on unstructured textual error descriptions.
- Automatic text classification and indexing, based on structured data, such as author, publication date, predefined keywords or categories (structured), and on the unstructured content of the texts themselves.

We solve our sample use case in two steps:

1. First, we use text analysis tools for extracting keywords that represent potential risk factors from the unstructured information and merge these text analysis results into the structured data.
2. Then we perform a data mining analysis on these combined data.

When describing the involved steps in more detail, we do it in a way that permits us to practically execute each step using the DB2 Warehouse Design Studio and one of its standard sample data. The sample data, heart.data, can be found in the folder dwe\im\IMinerX\V9.5\samples\ModelingDB2 on your DB2 Warehouse server. They can be loaded into a sample database using the script heartImport.db2, which also resides in that directory. To that purpose, open a DB2 command window (on MS Windows by selecting **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**) and enter the following:

```
cd c:\Program Files\dwe\im\IMinerX\V9.5\samples\ModelingDB2
db2 -tf heartImport.db2
```

10.2 Sample use case: cardiac diseases

In a medical study, 240 patients with cardiac diseases were monitored for mortality risk. For each patient, 13 columns of structured information were collected:

1. AGE: The patient's age.
2. SEX: The patient's gender (m/f).
3. PAIN_TYPE: An identifier between 1 and 4 describing the pain that was felt before or during the attack (1=angina, 2=abnormal angina, 3=nonanginal pain, 4=asymptomatic).
4. BLOOD_PRESSURE: Systolic blood pressure (in mmHg) at rest.
5. CHOLESTEROL: The total Cholesterol level (in mg/dL). A value above 240 is considered as problematically high.
6. ECG: An identifier describing the patient's resting electrocardiogram (ECG) pattern (0=normal, 1=abnormal, or 2=left ventricular hypertrophy)
7. HEART_RATE: Maximum heart rate (in pulses per minute) achieved in performance ECG.
8. ANGINA: Indicates whether or not the patient experiences angina pectoris as a result of physical exercise.
9. OLD_PEAK: Depression (relative to rest) of the so-called ST segment in the ECG induced by physical exercise.
10. SLOPE: Slope of the peak exercise ST segment (1=up, 2=flat, or 3=down).
11. NUM_VESSELS: Number of major vessels colored by fluoroscopy (which indicates a pathological state).
12. THAL: Result of Thallium 201 (201Tl) test: (3=normal, 6=fixed defect, or 7=reversible defect).

13. DISEASED: Indicates whether or not (y/n) the patient died of heart disease during the two years since the beginning of the study.

In addition, some unstructured textual information was collected for each patient. This included a short medical history with data on the patient's lifestyle and medical symptoms that might or might not be relevant risk factors for heart diseases.

A typical data record, when written in column separated values (csv) form, looks like the following:

59,"m","4",110,239,"2",142,"y",1.2,"2","1","7","y","smokes at least 20 cigarettes/day, physically inactive, alcoholic addict, adrenocortical obesity, often headache, apnea"

10.2.1 Frequent term analysis

As the first step of the text analysis, we perform a frequent term analysis on the available medical history texts, in order to discover which type of information we might expect to detect.

For that purpose, we create a new Warehousing project in the Design Studio and expand the corresponding entry in the Design Studio project explorer window. Right-clicking the sub-item **Frequent Terms** of the project item **Text Analysis** and selecting **New** → **Frequent Term Extraction** creates a new frequent term analysis instance. This is depicted in Figure 10-1.

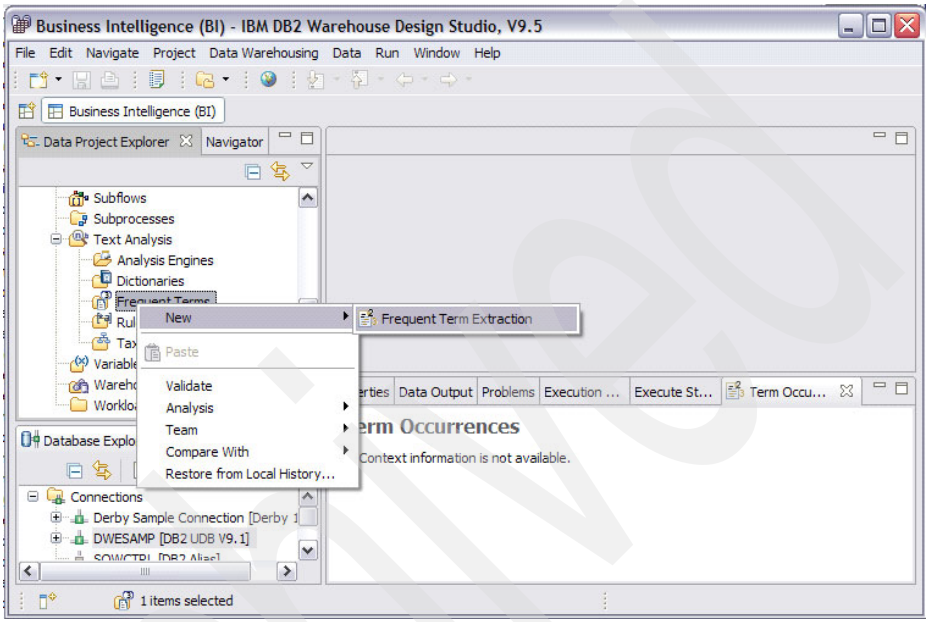


Figure 10-1 Frequent term analysis

A wizard then pops up in which we have to specify a name for the extraction result and a database connection. We select **Heart Keywords** as the name and our sample database as the connection. On the next wizard page, we select the table HEART, and the following wizard page allows us to select the data column that is to be analyzed. Per the default, the textual column with maximum length, MEDICAL_HISTORY, is selected. We accept this choice, and also the following default selection. On the next wizard page, we specify which types of terms are to be included in the analysis. We check the check boxes **Noun** and **Adjective - Noun**. Furthermore, we want to look for single adjectives. To that purpose, we select **New** → **Adjective** → **OK**, and then **Finish**. The resulting window is depicted in Figure 10-2.

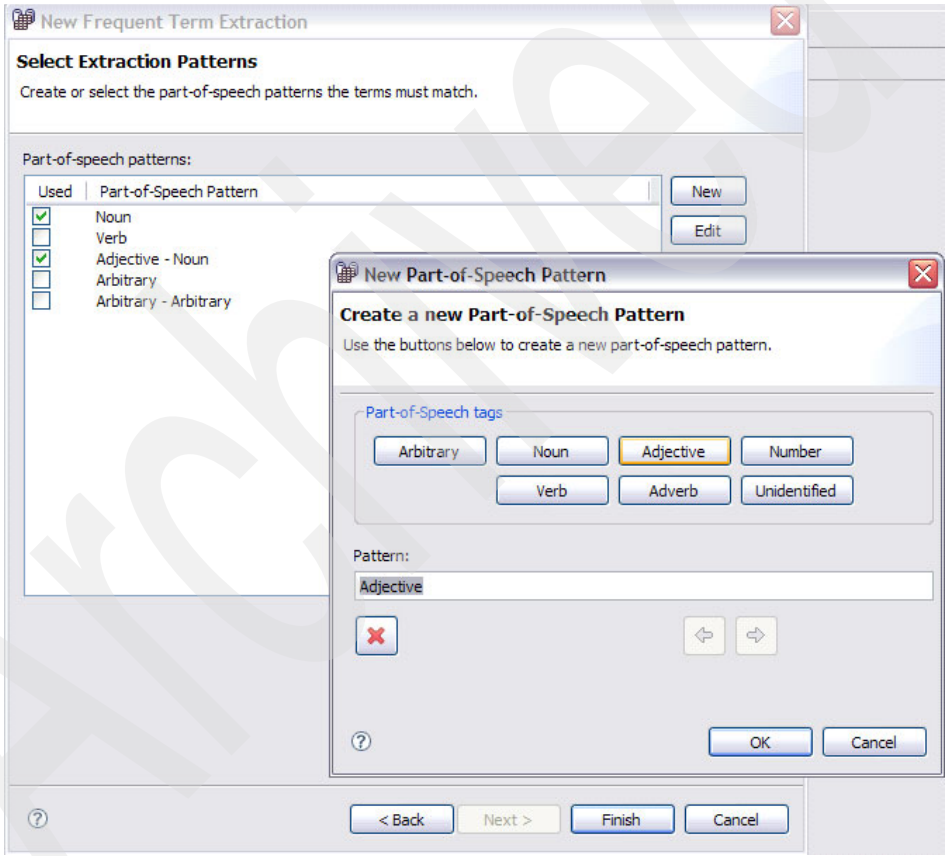


Figure 10-2 Select extraction patterns

Now the frequent term analysis automatically starts. Once it is finished, you can open the result by double-clicking the new sub-item **Heart Keywords** of the item **Frequent Terms** in the Data Project Explorer. The window is depicted in Figure 10-3.

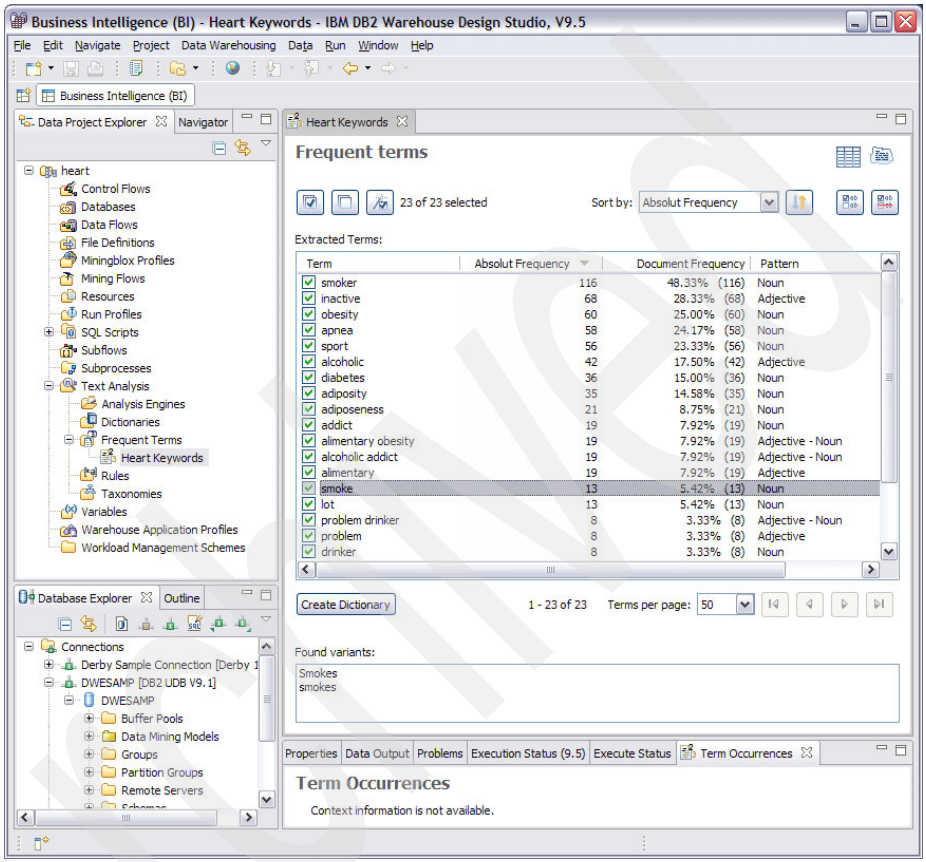


Figure 10-3 Frequent terms

The resulting lists of keywords contains, among others, some behaviors and properties that have been widely discussed as possible risk factors for cardiac diseases, such as:

- Physical inactivity
- Smoking
- Alcoholism
- Diabetes
- Stress
- Obesity
- Apnea, shortness of breath

For each keyword, the base form appears in the list. Several variants of the base form have been automatically associated with this base form. For example, the variants Smokes and smokes for the keyword smoke. In more sophisticated cases, the automatic base form matching does not work. For example, there are four unlinked base forms for obesity: obesity, adiposity, adiposeness, and alimentary obesity. You have to establish associations between these variants manually at a later stage.

10.2.2 Creating a dictionary

The next step in the analysis is to create a specific dictionary of keywords representing potential risk factors. Click *Create Dictionary* to create a dictionary out of the extracted frequent terms. As the name of the dictionary, we choose FactorDict. A new sub-item FactorDict of the item Dictionaries appears in the Data Project Explorer. Double-clicking it opens the content, as depicted in Figure 10-4.

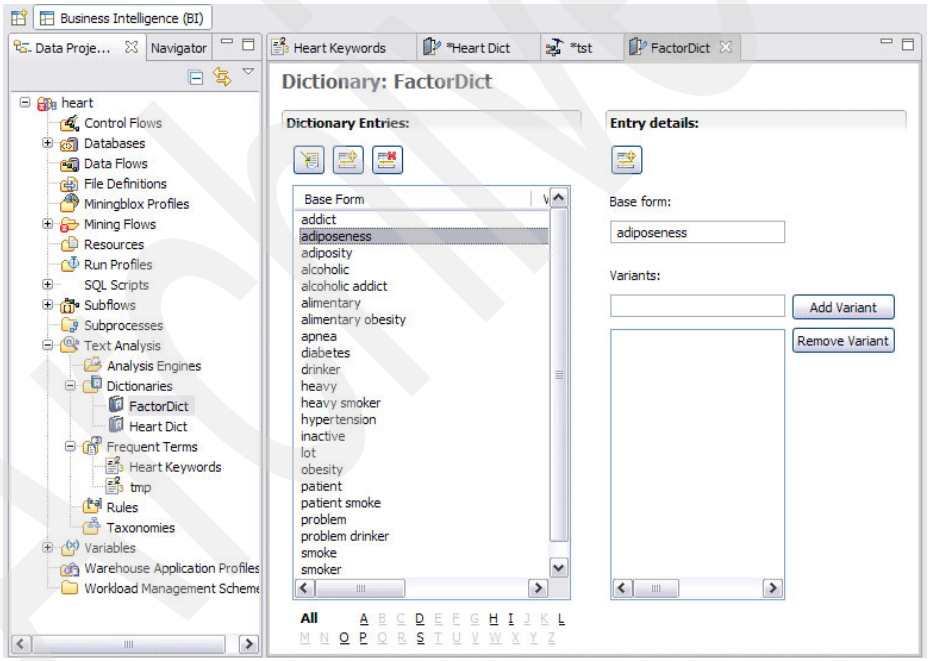


Figure 10-4 Dictionary

A first glance at this dictionary shows that it sill requires some manual editing. First, delete undesired items such as lot, patient, or problem. Then define some of the keywords as variants of a common base form. To do that, click the desired base form of the keyword in the keyword list and enter each desired variant into the free text field Variants, followed by clicking the **Add Variant** button.

You can also add completely new dictionary entries by clicking the icon above the keyword list that carries the yellow '+' symbol. After our editing, the dictionary could look like that depicted in Figure 10-5.

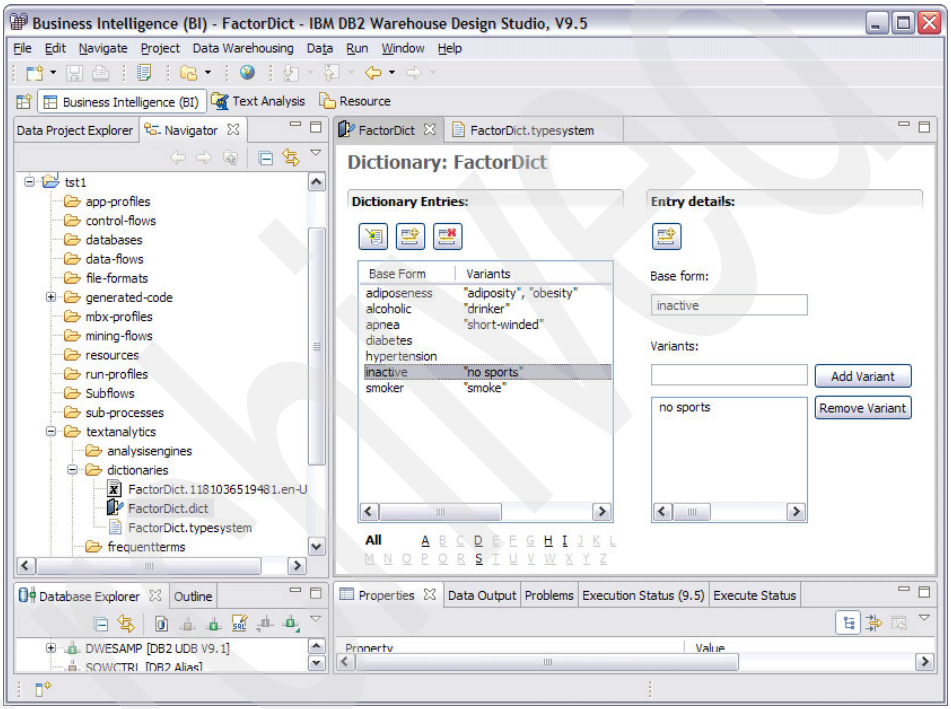


Figure 10-5 Adding variants

Dictionary based text analysis

Now, you can use the newly created dictionary for a dictionary based, or list based, text analysis. In each patient's data record store is a more structured form from the base forms of all risk factor keywords that are found in some variant of the patient's medical history.

For that purpose, create a new Mining Flow by selecting **File** → **New** → **MiningFlow** in the Design Studio main menu. Once you have clicked **Finish** in the Mining Flow creation wizard, a white canvas opens on which a mining flow can be modeled by drawing operators from the operator palette onto the canvas.

Draw a Table Source operator to the canvas (representing the table HEART), and then, a DictionaryLookup operator. In Properties view of this operator, configure the Analysis Results page. Select **FactorDict** as the dictionary and as the type of system. In the table of result columns, there are, by default, five entries. We are only interested in the entry FACTORDICT."baseform". This entry contains the base forms of all keywords that were found in the analyzed texts. The other four entries contain an internal ID of each occurrence, the occurrence's position in the text, and the actual variant of the keyword that was found in the text. Remove the other four entries. At the end, the Analysis Results page should have the content as depicted in Figure 10-6. There, the '+' button of the two operators has been clicked in order to expand the operators' contents.

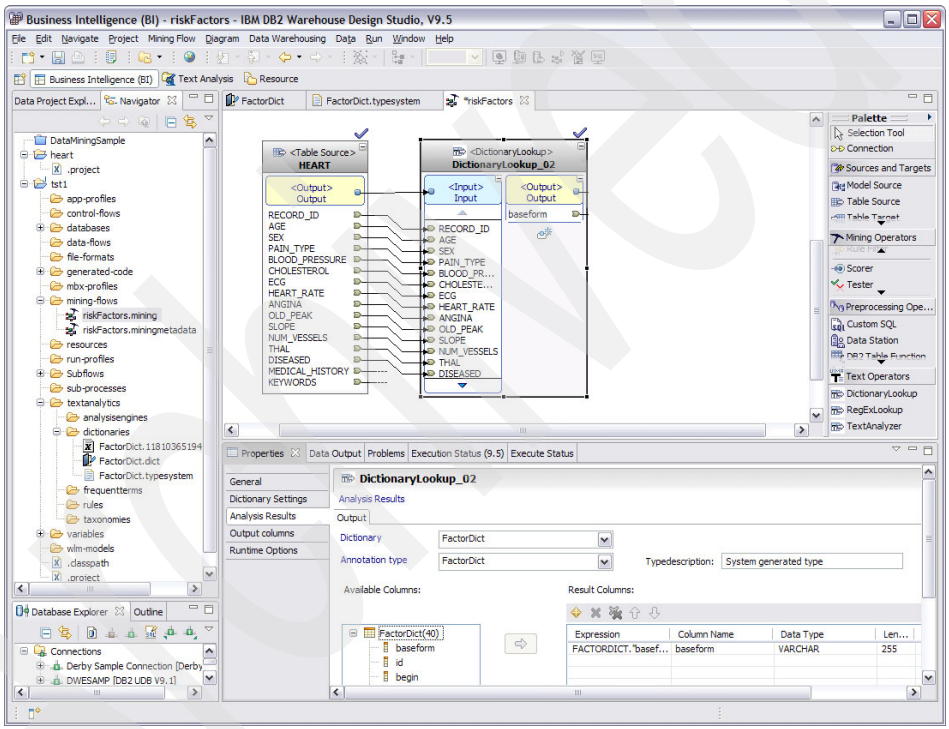


Figure 10-6 Analysis results

Now open the Output Columns page, select the column RECORD_ID and click the right arrow button. This adds the column RECORD_ID to the list of output columns of the dictionary lookup operator. Column RECORD_ID is needed as grouping information, which relates the extracted keywords to the records in which they were found.

Merging extracted keywords and structured data

The output of the dictionary lookup operator contains more than one row per patient record whenever more than one keyword is found in the medical history text. In this form, the text analysis results cannot be merged with the existing structured data columns of the table HEART. Therefore, an additional operator is needed, the Item Aggregator, for aggregating multiple rows per patient record into one single text analysis result. In the Column Properties page of the Item Aggregator, specify the column RECORD_ID as key column and the column baseform as set column. The result can be seen in Figure 10-7.

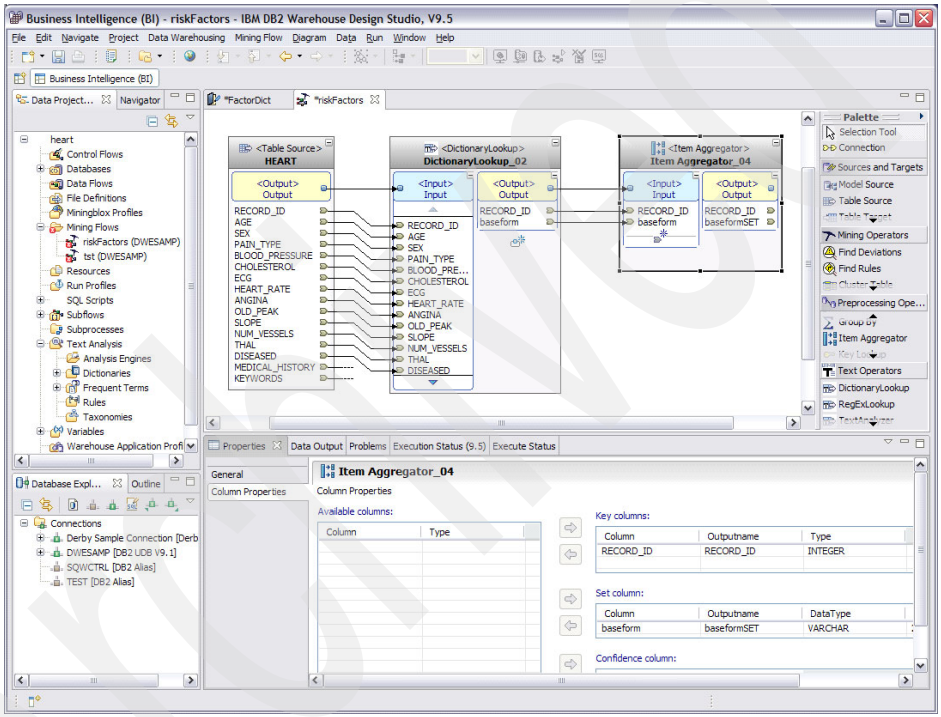


Figure 10-7 Item aggregator

The output of the Item Aggregator can now be joined with the initial table HEART, using the Table Join operator. Draw the Table Join operator to the canvas and connect its input ports to the outputs of the item aggregator and the original HEART table. In the Condition List page of the operator's Properties view, define the join condition. In this case, the two tables' RECORD_ID values should be equal. Either type this criterion into the free text field, or click the **...** button for opening the SQL Condition Builder. In the Condition Builder, compose the condition text by mouse clicks. Refer to the window depicted in Figure 10-8.

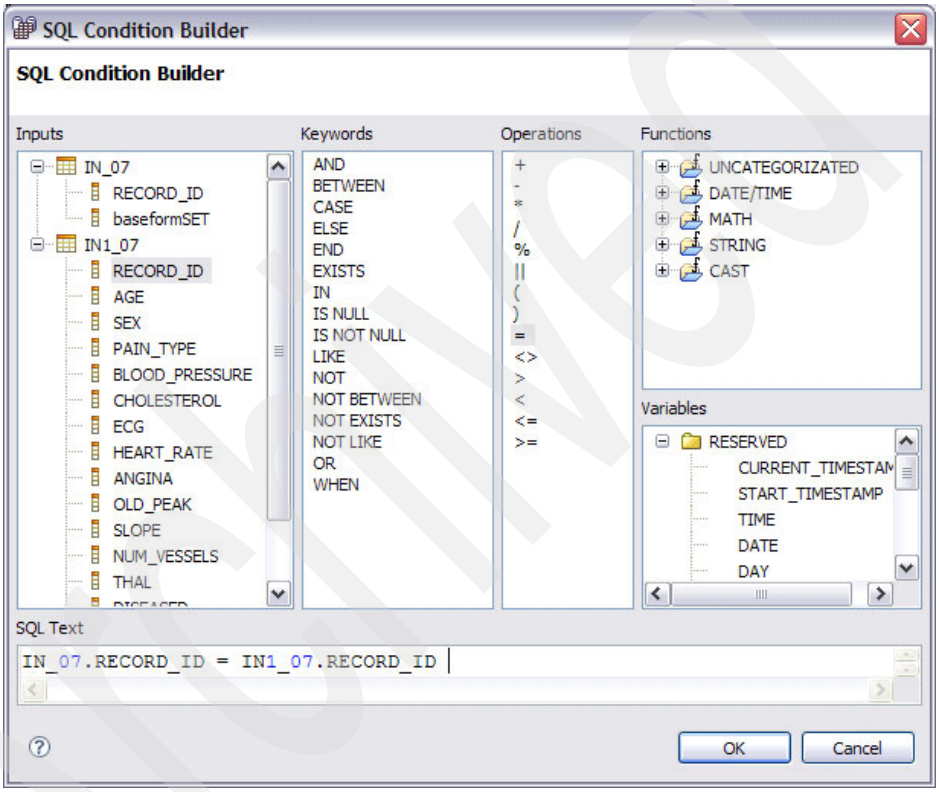


Figure 10-8 SQL condition builder

In the Select List page of the join operator's Properties view, delete one of the two RECORD_ID columns from the result list. The resulting mining flow has the form depicted in Figure 10-9.

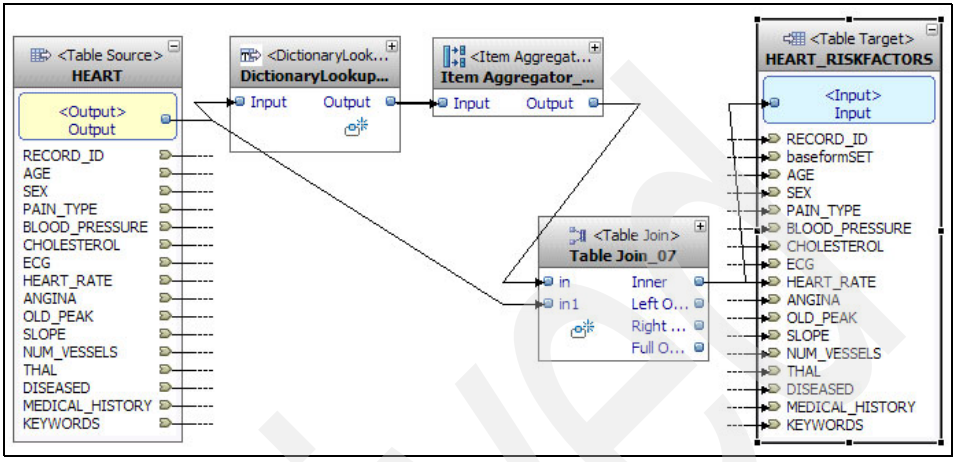


Figure 10-9 Mining flow

Having reached the first of two goals, the new table contains the results of a free-text analysis in the column baseformSet together with the original structured data columns. The XML format of the column baseformSet is suitable for being used within adjacent data mining flows.

Note: In order to make it possible to skip the text analysis steps when reproducing this use case on your computer, the original table HEART already contains a column KEYWORDS with the same content as the column baseformSET that was created in this section.

10.2.3 Data Mining Analysis on the enriched data

The table HEART has a data column called KEYWORDS. This column contains, for each patient, a set of potential risk factors that were detected by a previous text analysis of the patient's free-text medical history. The contents of the column KEYWORD is available in a format that can be exploited by the data mining component of DB2 Warehouse. A typical KEYWORDS value could look as follows:

```
<item>smoker</item><item>physical
inactivity</item><item>alcoholic</item> <item>obesity</item>
```

Altogether, each record of table HEART contains:

- A key column containing the record or patient ID
- The 13 structured data columns (AGE, SEX, …, DISEASED)
- The free text medical history column
- A KEYWORD column containing the extracted keywords in XML format

The next goal is to run a data mining analysis on the HEART data. To do this task, create a new mining flow in the Design Studio. In the white canvas window, start modeling the mining flow by dragging and dropping graphical operator icons from the palette window on the right side of the GUI. Choose the **Table Source operator**, drag it to the canvas and select the table named **HEART** in the pop-up window that asks for the table name.

Now explore the content of table HEART by right-clicking the table operator and then selecting **Data** → **Sample Content**. When doing so, Design Studio shows, in the lower part of the GUI, a Sample Contents window that looks like the one depicted in Figure 10-10.

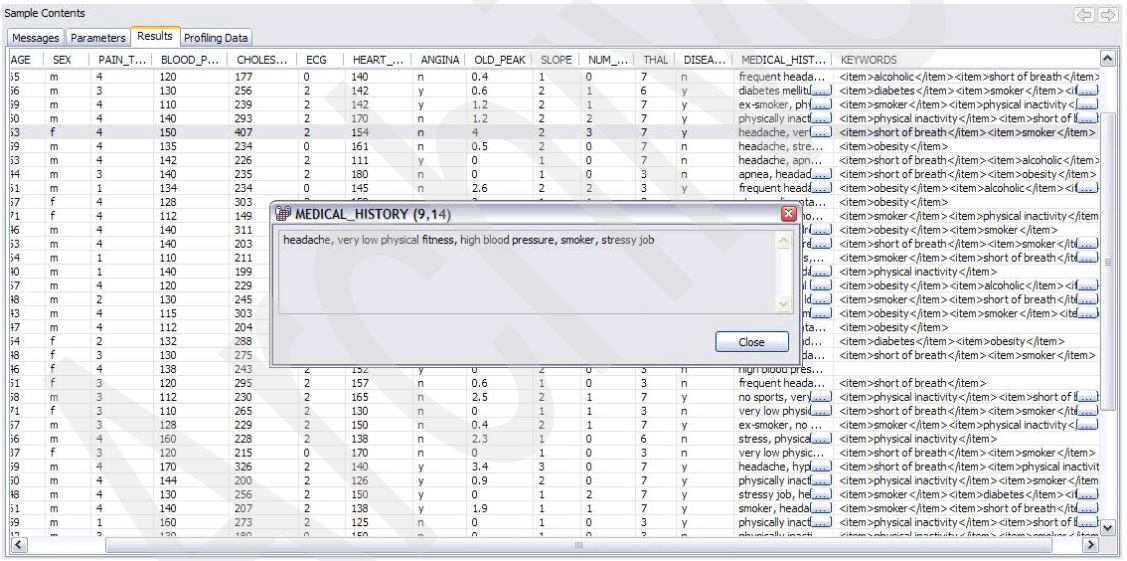


Figure 10-10 Sample contents window

The next goals are to build two mining models for the heart data:

1. An association rule model for detecting general risk factors for cardiac mortality
2. A prediction model for predicting the mortality risk for single patients with known medical history

The predictive power of a prediction model should always be tested on unknown data that was not used when the model was trained. Therefore, apply a Data Splitter operator on table HEART in order to split it in 75% training data and 25% test data.

In the palette of Preprocessing Operators on the right side of the GUI, find a *Splitter operator*, drag it to the canvas, and connect its input port to the output port of table HEART.

As the splitting criterion defines RECORD_ID <= 180 for the training data and RECORD_ID > 180 for the test data. The splitting criterion can be entered from the Condition List page in the Properties view, which opens up in the lower part of the GUI if you click the Splitter operator on the canvas. By clicking the **[...]** button in the activated row of the condition list page, you can start an SQL editor for composing the condition text. The result is depicted in Figure 10-11.

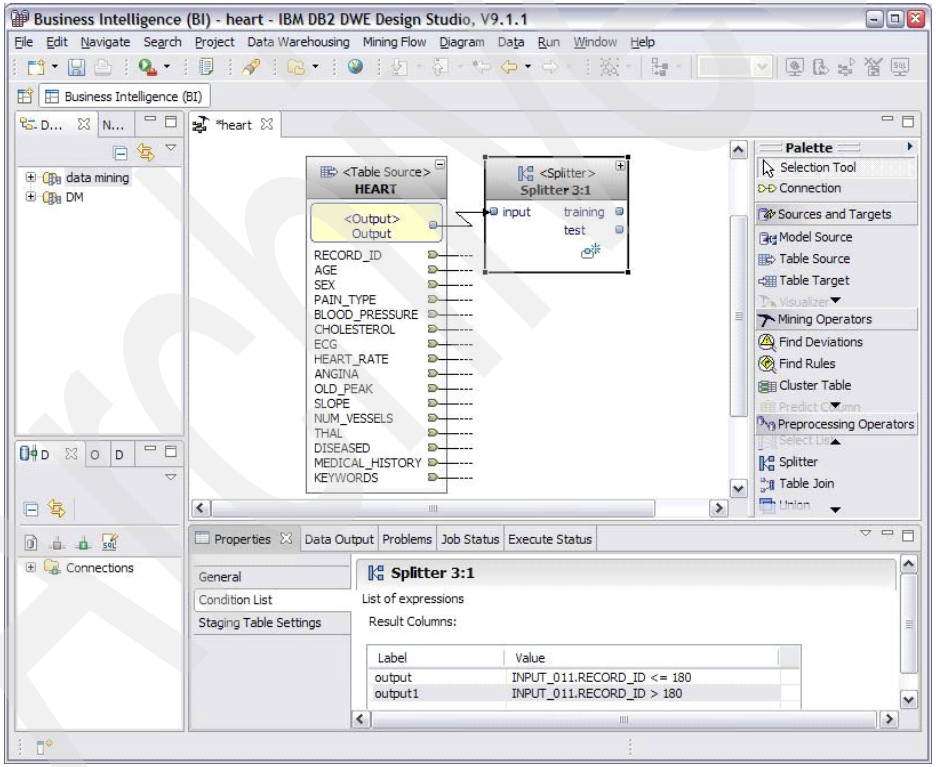


Figure 10-11 Splitter operator

Now complete the two mining tasks by successively adding an Associations, Predictor, Tester, and two Visualizer operators to the mining flow. At the end, there should be a picture such as that depicted in Figure 10-12.

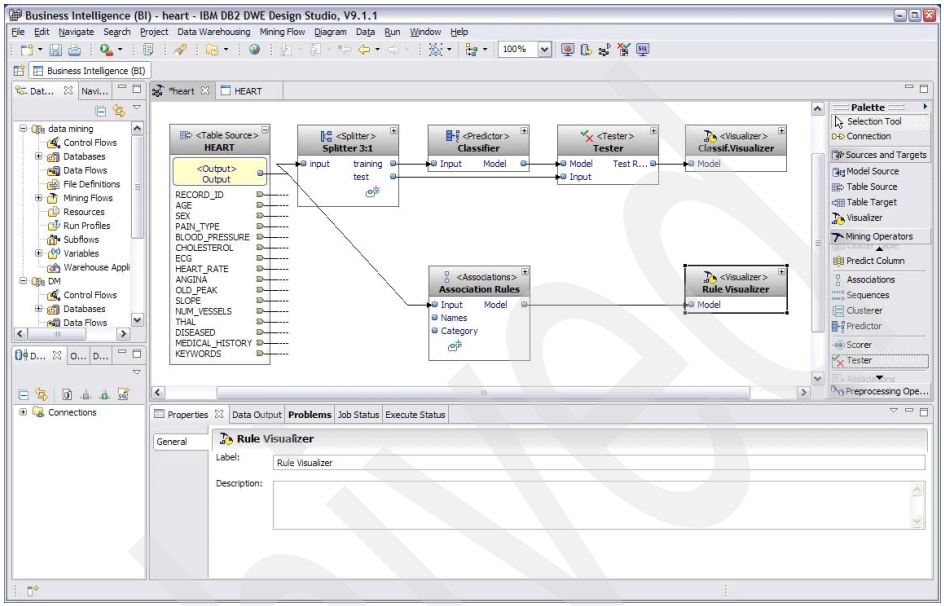


Figure 10-12 Mining flow

10.2.4 Mining results: risk factor analysis

Before building an Association Rule model, you should get a first insight into the correlations between mortality and the collected patient data. For that purpose, right-click the table source HEART and select **Value Distributions** → **Multivariate**. In the upper part of the GUI there is a list of all columns of table HEART and their statistics (such as mean, minimum, and maximum for the numeric columns, and text length statistics for the textual columns). By clicking any of these rows (keep the Ctrl key pressed while clicking to select multiple rows), you can open a histogram representation of the corresponding data column in the lower part of the GUI.

In each of these histograms, you can now select a subset of all bars by clicking those bars to be active. Instantaneously, the content of all other histograms change and show the distribution of the remaining active data records. For example, Figure 10-13 shows the value distributions when only the diseased patients' records are activated.

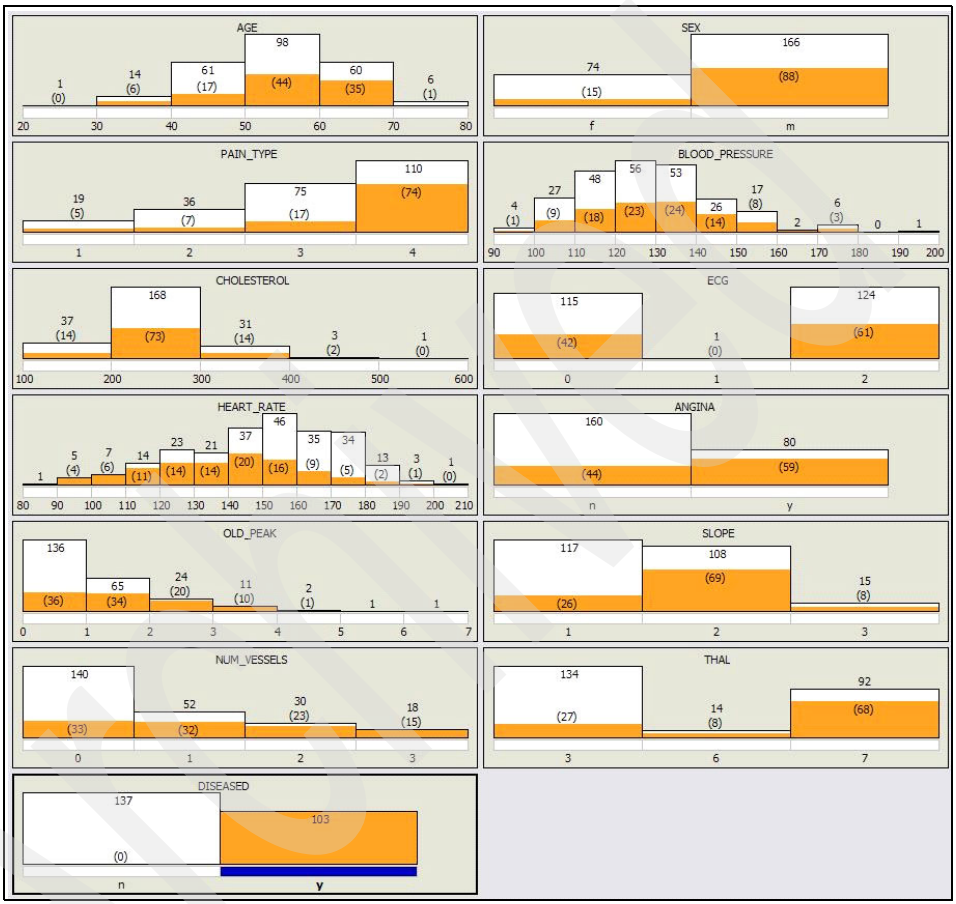


Figure 10-13 Value distributions for diseased patient

By also clicking the *m* bar of the histogram for data field SEX, you can further narrow down the active records to the diseased male patients, and so on.

What information does this tell us about mortality risk factors? It tells us that people with maximum HEART_RATE below 120, OLD_PEAK of 2 or larger, or NUM_VESSELS of 2 or larger have a dramatically increased risk. Also, the facts AGE>60, SEX=m, PAIN_TYPE=4, ANGINA=y, SLOPE>1, or THAL>3 do significantly increase the mortality risk.

Also, maybe somewhat surprisingly, the ECG patterns and CHOLESTEROL or BLOOD_PRESSURE values do not have a significant impact on imminent mortality (whereas it is know from other studies that these values do have an impact for developing a coronary heart disease), for the patient group which is examined here.

In summary, the first date exploration has provided interesting insights, but a few of questions are still open:

1. Can the risk of the increasing effects of the various factors be quantified?
2. Are there certain combinations of several factors that only in combination have a risk-increasing effect?
3. Can the textual information be exploited? So far only the structured data has been used, which does not include lifestyle and medical history factors.

In order to answer these questions, an Association Rule model will be created. But before starting the model training, there are a few modifications to be performed in the various property pages of the Associations operator. On the Mining Settings page:

- Increase the maximum rule length from 2 to 3 (to find risk factor combinations, such as rules with two body items and one head item).
- Increase the minimum confidence to 50%.
- Increase the minimum support to 5% (which means that each rule found must be supported by at least 12 out of the 240 data records).
- Specify that the values (for numeric columns such as HEART_RATE) will be discretized into 4 bins or value ranges.
- Define medically reasonable interval ranges (in the Optional Parameters text field), which redefine the default bin boundaries for the numeric columns.

The results are depicted in Figure 10-14.

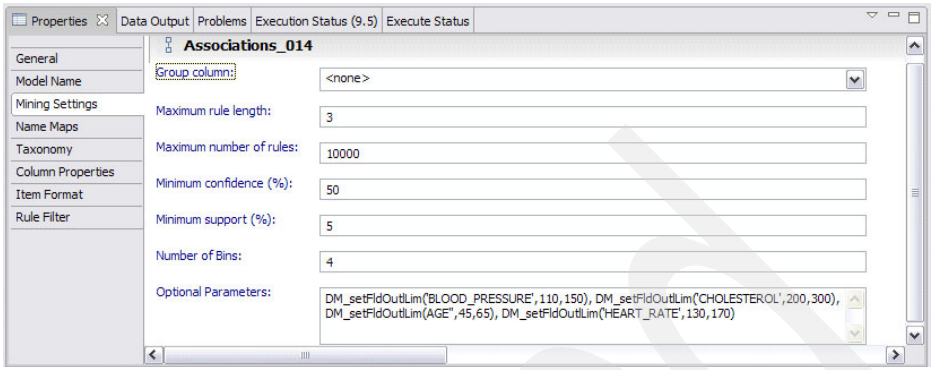


Figure 10-14 Default properties

In the Column Properties window, the columns RECORD_ID and MEDICAL_HISTORY for the mining are deactivated, and the column KEYWORDS are specified as set valued. That results in the window depicted in Figure 10-15.

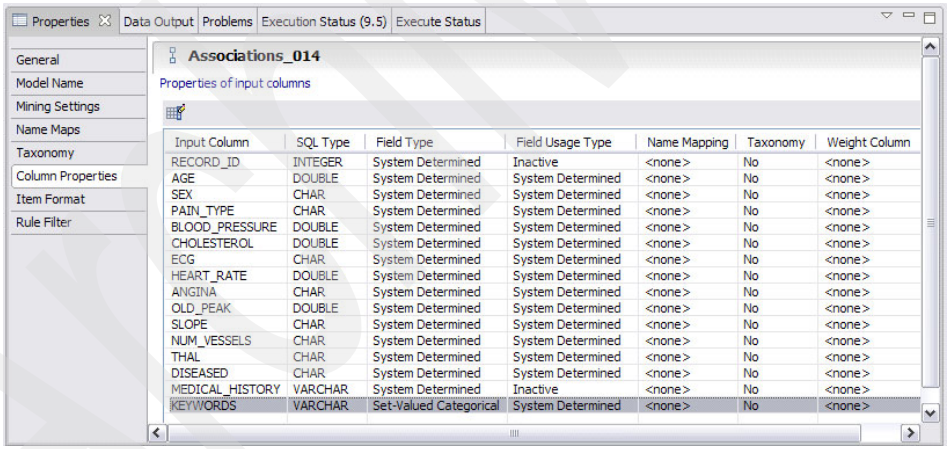


Figure 10-15 Setting keywords

Finally, two important additional rule filters are defined in the Rule Filter property page:

1. Specify a range constraint filter ‘Lift must not be between 0.8 and 1.2’. A lift value close to 1 for a rule A'B means that p(B) = p(BlA). In other words, the occurrence of A does not have any influence on the occurrence probability of B. Those uninteresting rules should not be found in the resulting model.
2. Specify an item constraint filter 'item DISEASED=y must be in the rule head'. In looking for risk factors for mortality, rules which have 'DISEASED=y' on the right hand side of the rule are the only ones in which there is an interest.

After performing these steps, the Rule Filter property page should look like Figure 10-16.

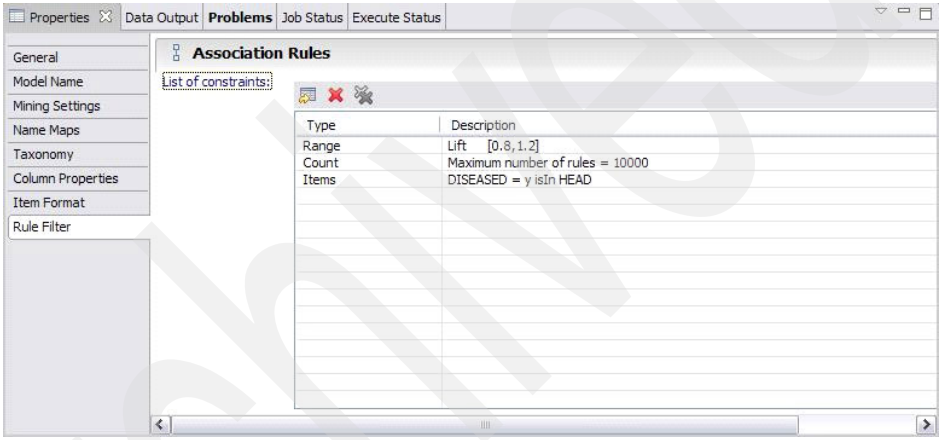
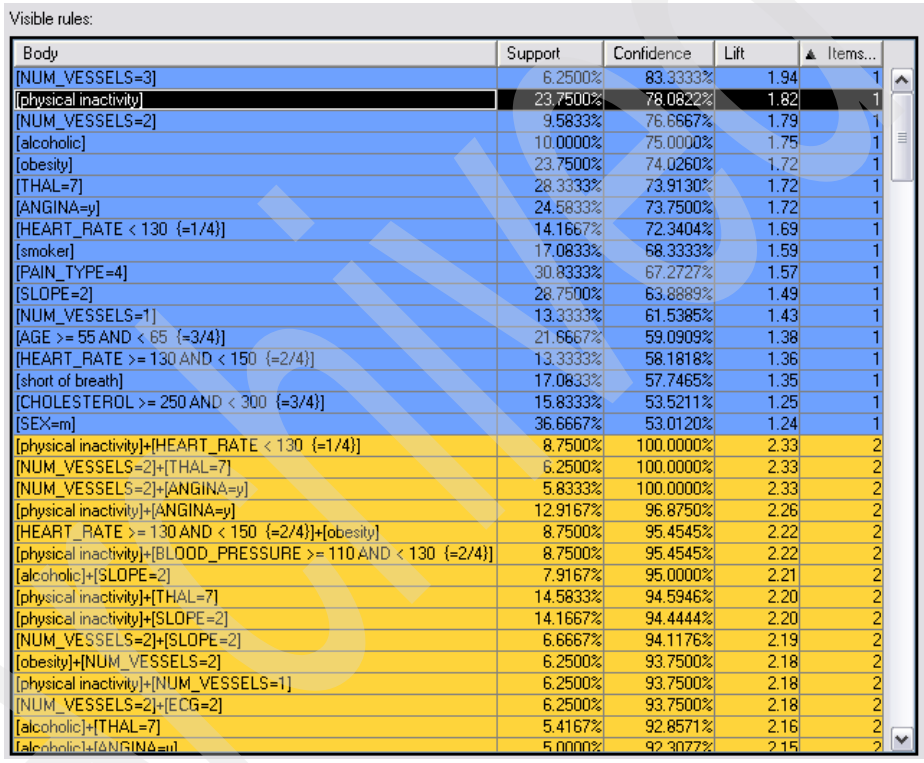


Figure 10-16 Rules properties

Now all preparations are done and the mining process can be started by right-clicking the **Rule Visualizer** operator, and then selecting **Run to this step**. After a few seconds, the associations visualizer pops up and shows a tabular view of the detected rules, as shown in Figure 10-17. Your tabular view might look slightly different, having more columns and a different sorting order. You can modify that app earance using the following menu buttons in the associations visualizer:

- **File → Visualizer Properties → Rules**
- **File → Visualizer Properties → Rules Columns**
- **File → Visualizer Properties → Rules Sort**


Visible rules:

Body	Support	Confidence	Lift	▲ Items...
[NUM_VESSELS=3]	6.2500%	83.3333%	1.94	1
[physical inactivity]	23.7500%	78.0822%	1.82	1
[NUM_VESSELS=2]	9.5833%	76.6667%	1.79	1
[alcoholic]	10.0000%	75.0000%	1.75	1
[obesity]	23.7500%	74.0260%	1.72	1
[THAL=7]	28.3333%	73.9130%	1.72	1
[ANGINA=y]	24.5833%	73.7500%	1.72	1
[HEART_RATE < 130 {=1/4}]	14.1667%	72.3404%	1.69	1
[smoker]	17.0833%	68.3333%	1.59	1
[PAIN_TYPE=4]	30.8333%	67.2727%	1.57	1
[SLOPE=2]	28.7500%	63.8889%	1.49	1
[NUM_VESSELS=1]	13.3333%	61.5385%	1.43	1
[AGE >= 55 AND < 65 {=3/4}]	21.6667%	59.0909%	1.38	1
[HEART_RATE >= 130 AND < 150 {=2/4}]	13.3333%	58.1818%	1.36	1
[short of breath]	17.0833%	57.7465%	1.35	1
[CHOLESTEROL >= 250 AND < 300 {=3/4}]	15.8333%	53.5211%	1.25	1
[SEX=m]	36.6667%	53.0120%	1.24	1
[physical inactivity]+[HEART_RATE < 130 {=1/4}]	8.7500%	100.0000%	2.33	2
[NUM_VESSELS=2]+[THAL=7]	6.2500%	100.0000%	2.33	2
[NUM_VESSELS=2]+[ANGINA=y]	5.8333%	100.0000%	2.33	2
[physical inactivity]+[ANGINA=y]	12.9167%	96.8750%	2.26	2
[HEART_RATE >= 130 AND < 150 {=2/4}]+[obesity]	8.7500%	95.4545%	2.22	2
[physical inactivity]+[BLOOD_PRESSURE >= 110 AND < 130 {=2/4}]	8.7500%	95.4545%	2.22	2
[alcoholic]+[SLOPE=2]	7.9167%	95.0000%	2.21	2
[physical inactivity]+[THAL=7]	14.5833%	94.5946%	2.20	2
[physical inactivity]+[SLOPE=2]	14.1667%	94.4444%	2.20	2
[NUM_VESSELS=2]+[SLOPE=2]	6.6667%	94.1176%	2.19	2
[obesity]+[NUM_VESSELS=2]	6.2500%	93.7500%	2.18	2
[physical inactivity]+[NUM_VESSELS=1]	6.2500%	93.7500%	2.18	2
[NUM_VESSELS=2]+[ECG=2]	6.2500%	93.7500%	2.18	2
[alcoholic]+[THAL=7]	5.4167%	92.8571%	2.16	2
[alcoholic]+[ANGINA=y]	5.0000%	92.3077%	2.15	2

Figure 10-17 Rule visualizer

Is it now possible to answer the three following questions, which led to the creation of the model?

1. Can you quantify the risk increasing effects of the various potential factors?

   Yes. The body parts of the blue rules in the picture (that is, the length-2 rules) are the significant single risk factors detected in the data. Each rule's Lift is the corresponding factor's mortality augmentation effect. For example, the highlighted factor “physical inactivity” increases the mortality risk by 1.82 or 82%. Furthermore, learning that the factor was found with 23.75% of the patients, and 78.08% of these patients actually died.

2. Are there certain combinations of several factors which, only in their combination, have a risk-increasing effect?

   No. None of the length-3 rules has a lift factor larger than 2.33, and each length-3 rule's lift has roughly the value that could be expected by adding the effects of the two separate risk factors.

3. Have you also exploited the textual information?

   Yes. You found that physical inactivity, alcoholism, obesity, smoking, and shortness of breath are in fact significant factors for mortality. On the other side, diabetes does not increase mortality.

10.2.5 Mining analysis: build and validate a prediction model

The next goal is to create a prediction model that can, in the future, be applied to predict the mortality risk for newly arriving patients with coronary heart diseases. Having such a model that works reliably could have tremendous benefits for these patients because intensive (and expensive) regular medical tests, controls, and treatments can be concentrated on those patients who really need it, and other patients can improve their quality of life by getting a positive prognosis and by reducing the time they spent with doctors or in hospital.

Fortunately, the Predictor operator, depicted in Figure 10-18, does not require as much fine-tuning as the Associations operator. Here, simply specify the:

- Target field of the prediction (field DISEASED).
- Mining algorithm: Choose the Naïve Bayes method, because it is best suited for dealing with set-valued content extracted from text.
- Content of field KEYWORDS, which should be interpreted as a set of single values and not as one single string.

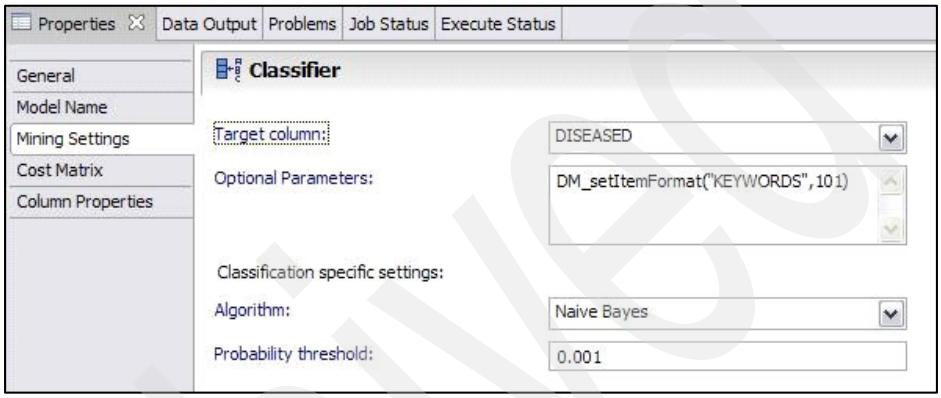


Figure 10-18 Predictor operator

After these adjustments have been done, start the entire model training and testing process by right-clicking the classification visualizer operator and then selecting **Run to this step**.

When the classification visualizer pops up (see Figure 10-19 on page 304), the first thing to appear is the model quality overview page. This page contains an overall rating of the model quality, and three detailed quality measures, from which the overall quality is calculated. They are:

1. Accuracy Quality: Measures the predictive accuracy of the model. The measure is defined as the number of correct predictions divided by the total number of predictions.
2. Ranking Quality: Measures the model capability to correctly separate the different target field values. The measure is defined as the ratio between the gain of the actual model and the gain of the best possible model.
3. Reliability Quality: Measures how much the model predictive accuracy on unknown data deteriorates compared to its predictive accuracy on data that was used to train the model.

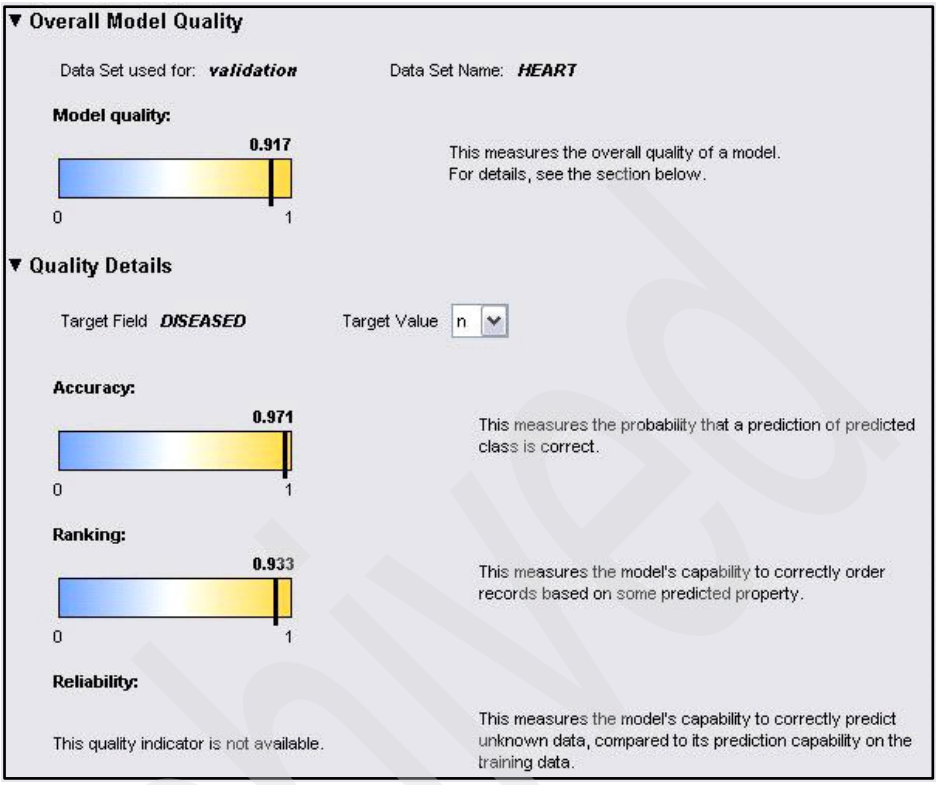


Figure 10-19 Classification visualizer

It appears that the model is very good. All quality values are close to their optimum value 1. The unstructured part of the training data, the field KEYWORDS, contributes significantly to this good model quality. This is depicted in Figure 10-20 on page 305. When KEYWORDS is deactivated as a mining field, the resulting model has considerably lower quality measures.

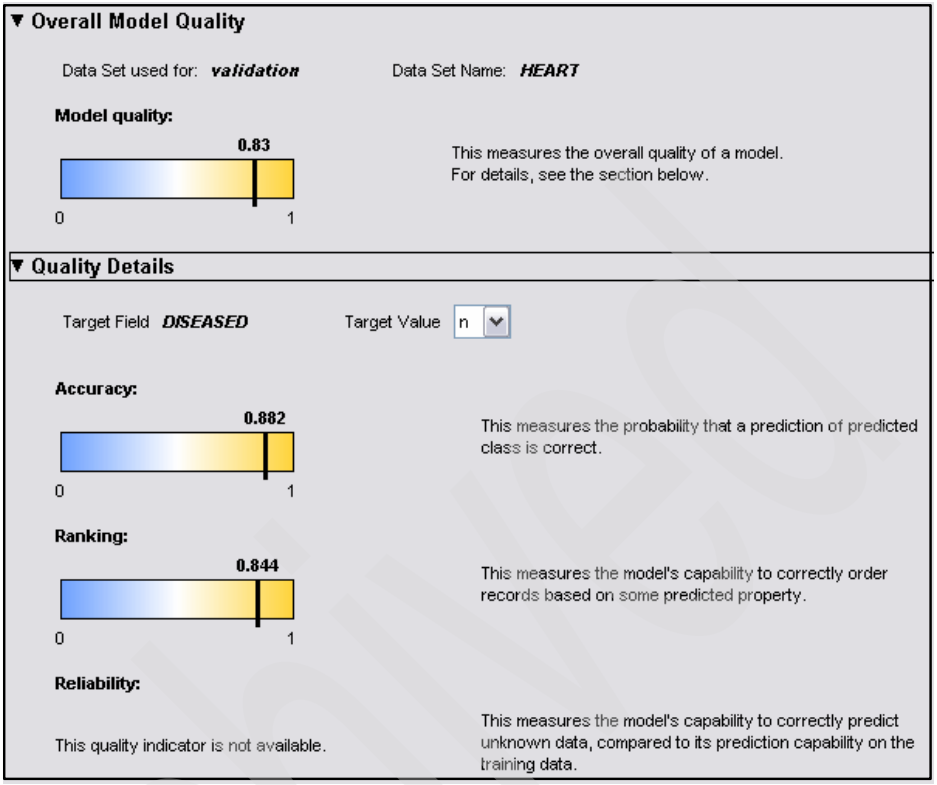


Figure 10-20 Model Quality

Another view of the classification visualizer is called the gains chart. There is a separate gains chart for each target field value T. In this case, the values T=y and T=n. In the gains chart, the x-axis carries all training data records ordered by decreasing predicted probability that the target field value is the chart's selected value T. On the y-axis, the accumulated number of all records with x-values between 0 and x is traced that actually have the target field value T. A useless model produces a straight line in the gains chart. The better the model performs in separating the records with target value T from all other records, the more the model's gains chart deviates from this straight line.

Figure 10-21 shows the created model's gains chart for the target value y. For comparison, the graph also shows a randomly predicting model's line and an optimally predicting model's line. From the gains chart, it can be seen that the model perfectly identifies about 33% of the 44% records with target value y, and also 40% of the 56% records with target value n. The remaining 27% of the records seem to be untypical cases that the model cannot classify with certitude.

The chart also shows the gains chart for a classification model that has been trained without using the unstructured information from field KEYWORDS.

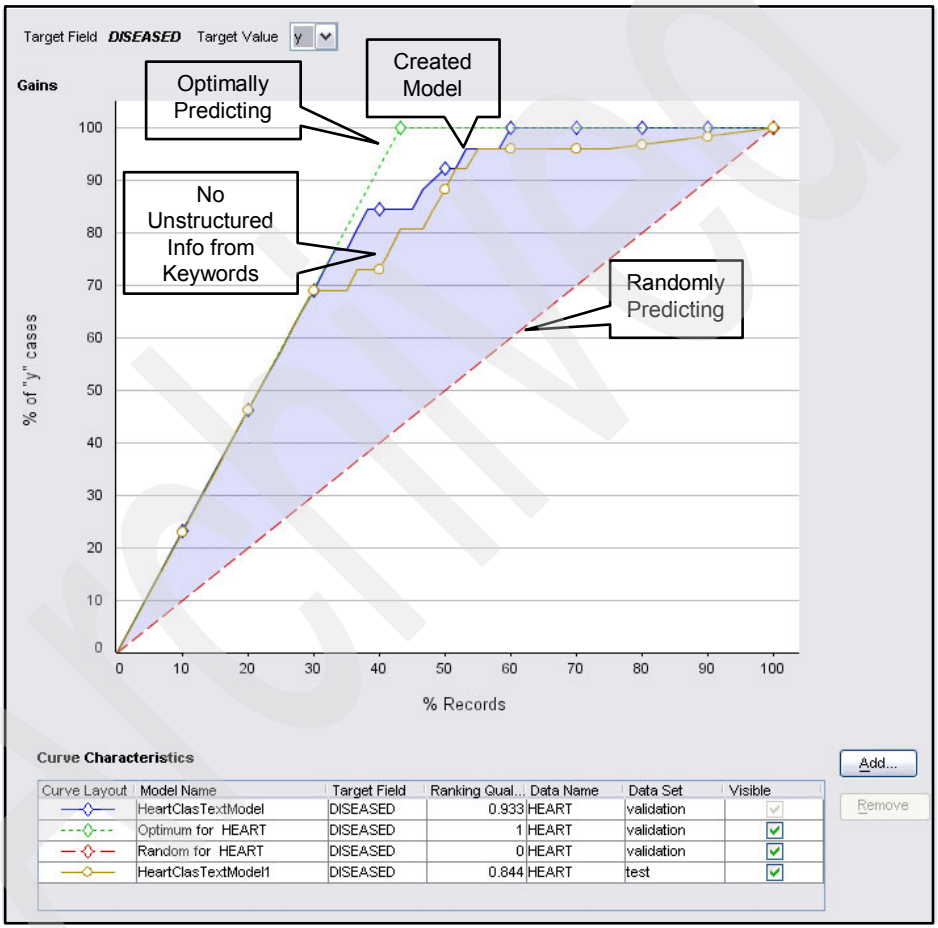


Figure 10-21 Gains chart

From the third view of the classification visualizer, the field importance view, you can see how much each of the mining fields has contributed to the predictive power of the model. This is depicted in Figure 10-22.

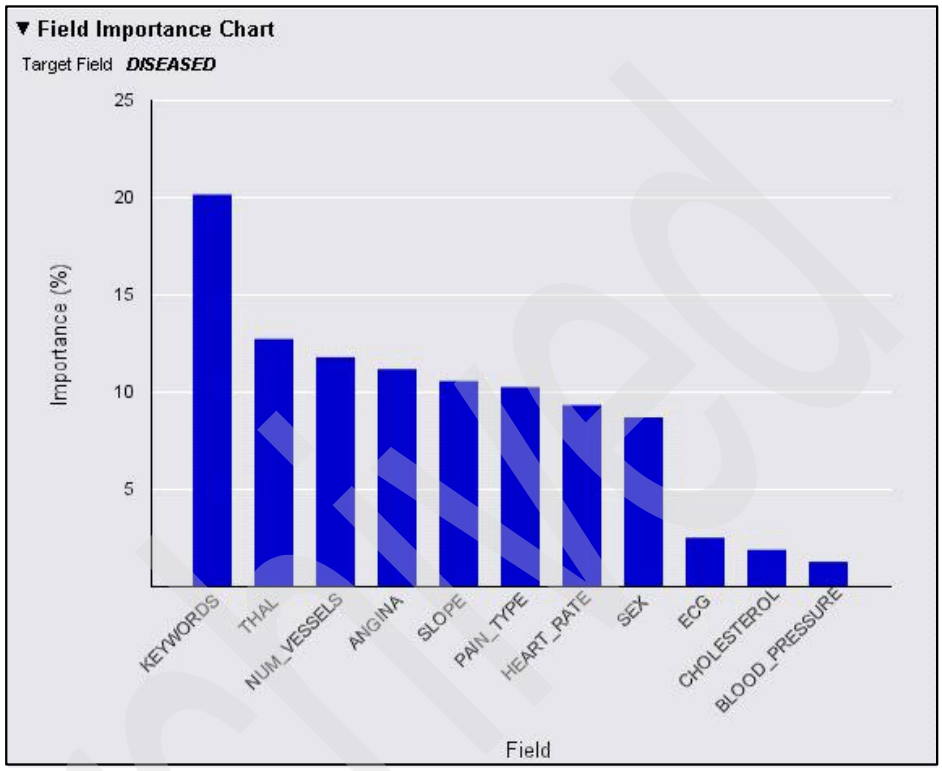


Figure 10-22 Field importance view

The chart confirms the first, qualitative impression obtained from the multivariate analysis of the training data. The fields ECG, CHOLESTEROL, and BLOOD_PRESSURE are not very useful for the prediction of mortality, whereas the other structured data fields are all relevant risk factors that significantly contribute to the predictive power of the model. Furthermore, you can see that the unstructured information, condensed into the set-valued field KEYWORDS, significantly improves the model. That field is the most important single contributor to the model's quality.

Summary

Combining structured data with information extracted from text, you have detected several risk factors for mortality of coronary heart disease patients, and have quantified the impact of each of these factors on the mortality risk.

In a second step, you have created a very powerful prediction model for the mortality risk. The model is able to classify about 40% of the patients with almost 100% certitude into the very low risk group, and 33% of the patients into the very high risk group.

Both kinds of insights could be leveraged in medical treatment practice, resulting in better targeted individual treatment plans, higher life quality and life expectancy for the patients, and reduced treatment costs.

10.3 Combining text mining and OLAP

In this section, we demonstrate how to use the text mining capabilities of DB2 Warehouse operators to transform unstructured data into a usable relational table that can be incorporated into an OLAP cube and into Alphablox reports.

In the sample scenario, an IT company called NetroSystem, Inc., wants to know which skills and knowledge are requested by their competitors (for example, which programming languages). Assume that this company has a collection of job offerings from eight others corporations in a database: Company1, Company2, Company3, Company4, Company5, Company6, Company7, and Company8.

However, none of the existing reporting tools are able to use this unstructured text-form data. The company's aim is therefore to obtain a structured form, which can be analyzed. In order to achieve this goal, we use the Dictionary Lookup Operator.

By using a user-defined dictionary, this operator will create annotations, and will find every occurrence of terms previously defined and mark their positions. As a result, we obtain a table that gives us the dictionary's terms found for each offer.

From this table, we will first design a star schema, and then build an OLAP cube. Finally, we will use this cube in an ALPHABLOX® based report to show the assets of the analysis.

10.3.1 Creating and using a dictionary

In this lesson, we define a mining flow that uses the Dictionary Lookup operator to extract the skills keywords from the job offerings. For that purpose, we design a new dictionary that describes a range of IT skills.

There is a table, called JOBS, which contains four columns:

1. COMPANY_NAME: The name of the company
2. TIME: Offer date
3. ID: A number that identifies the offer
4. JOB_DESC: The text-form job description

To create a new dictionary, first create a new warehousing project within DB2 Warehouse. In the Data Explorer View, expand the entry representing the project. Right-click the **Dictionaries** sub-item of the Text Analysis item and select **New** → **Dictionary**. Choose the name it_skills for the new dictionary. A dictionary editor appears as shown in Figure 10-23 on page 310. Note that we have only shown a partial view of the dictionary for readability.

A view of the dictionary is displayed on the left side of the editor, and includes all terms and a sample of their variants. On the right side is the editing zone (not shown in Figure 10-23 on page 310). You can add the terms and their variants. Note that for the common words, there is no need to enter a case variant. For example, if the word database is entered, the operator will also find terms such as Database, DataBase, and databases. On the contrary, for acronyms or special terms, it depends on the case in which the word is entered into the dictionary. To have the more efficient result, it is mandatory to enter the word in all lower-case (for example, if you enter j2ee, the dictionary will find J2EE™, but the opposite is not true).

Figure 10-23 shows the IT skills dictionary after all desired IT skills keywords have been entered.

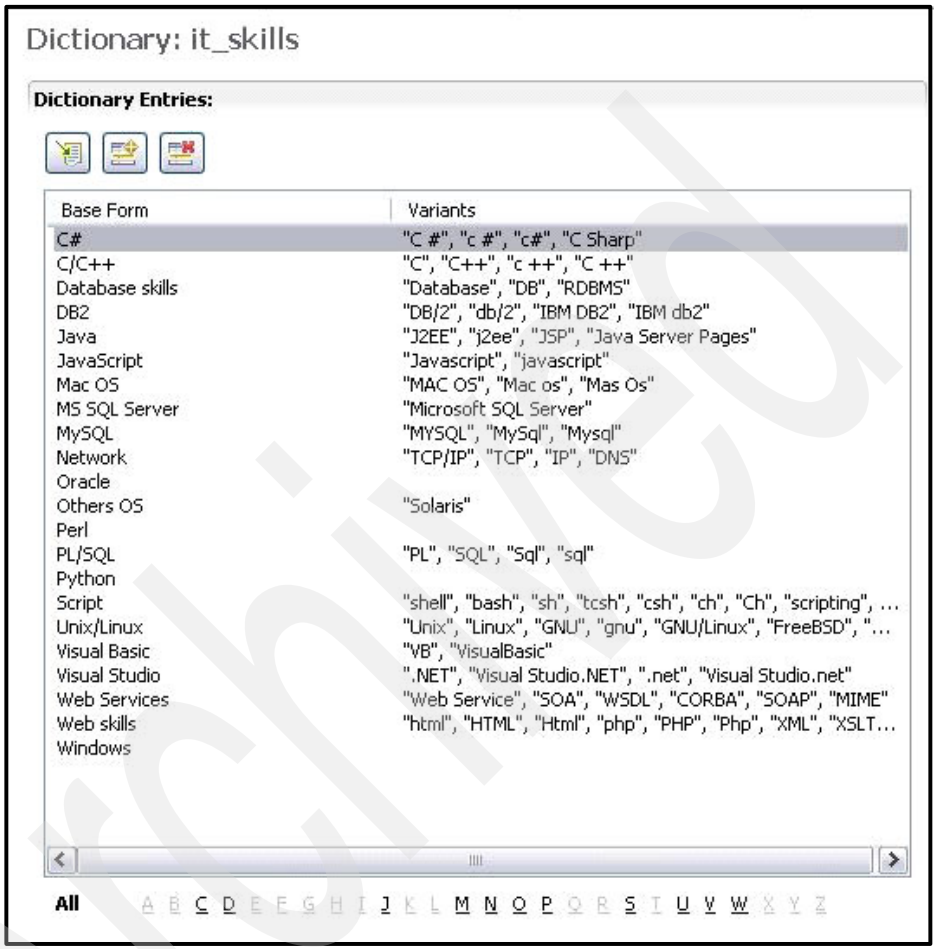


Figure 10-23 Dictionary editor (partial view)

Now we create a new mining flow (by selecting **File** → **New** → **Mining Flow**) and place a Table Source operator on it that represents the table JOBS. To the output port of this table operator we connect a DictionaryLookup operator. On the Dictionary Settings page of this operator's Properties view, we select **JOB_DESC** as the input text column and **English** as the language. On the Analysis Results tab, we choose it_skills as the dictionary and as the annotation type. For Result Columns, we select baseform and coveredText. We rename these two columns SKILLS_CAT and SKILLS_DETAILS respectively. On the Output Column page, we choose COMPANY_NAME, TIME, and ID.

Finally, we right-click the output port of the Dictionary Lookup Operator, and choose **Create suitable table**. If we name the new table IT_SKILLS_ASKED, our flow will look like that depicted in Figure 10-24.

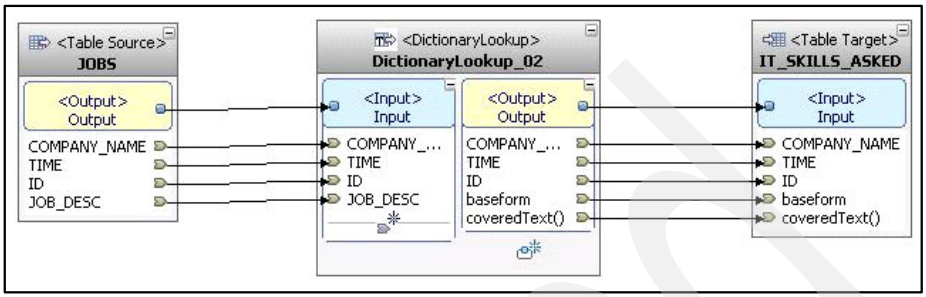


Figure 10-24 New mining flow

After having executed this flow, we can inspect the content of the generated table by right-clicking the table target operator and choosing **Sample Contents**. The contents will be similar to those depicted in Table 10-1.

Table 10-1 Generated table contents

Company Name	Time_1	ID	Skills Cat	Skills Detail
Company1	2005_03_02	3	Database	Database
Company1	2005_03_02	6	Database	Database
Company1	2005_03_02	21	C/C++	C
Company1	2005_03_02	21	Oracle®	Oracle
Company1	2005_03_02	21	Oracle	Oracle
Company1	2005_03_02	23	C/C++	C
Company1	2005_03_02	28	Windows	Windows
Company1	2005_03_02	28	UNIX/Linux	Linux
Company1	2005_03_02	39	Network	Network
Company1	2005_03_02	45	Network	Network
Company1	2005_03_02	46	Network	Networking
Company1	2005_03_02	49	C/C++	C
Company2	2006_10_19	8151	DB2	DB2
Company2	2006_10_19	8151	C/C++	C

Company Name	Time_1	ID	Skills Cat	Skills Detail
Company2	2006_10_19	8151	C/C++	C
Company2	2006_10_19	8151	DB2	DB2
Company2	2006_10_19	8154	C/C++	C
Company2	2006_10_19	8154	C/C++	C
Company2	2006_10_19	8154	C/C++	C
Company3	2007_01_09	9211	Other OS	Solaris™
Company3	2007_01_09	9211	UNIX/Linux	Linux
Company3	2007_01_09	9211	Network	Networking
Company3	2007_01_09	9211	UNIX/Linux	Linux
Company3	2007_01_09	9211	Windows	Windows
Company3	2007_01_09	9211	Other OS	Solaris
Company3	2007_01_09	9211	C/C++	C
Company3	2007_01_09	9211	C/C++	C+
Company3	2007_01_09	9211	C/C++	C
Company3	2007_01_09	9211	Java	Java
Company3	2007_01_09	9211	Script	Scripting
Company3	2007_01_09	9211	Perl	Perl

As can be seen, the obtained table links informative data about the offers (which company, when, and which offer) with the required analysis data, such as the IT skills asked for in these offers. This table can now be used for reporting. Here are some examples:

- Determine which skills are needed on the market, and consequently have an idea of the development lines from the other companies.
- Study the evolution for a time period.
- Observe which companies have the same interest as yours.

10.3.2 Building a Star Schema and an OLAP Cube

In this section, we create a star schema required to build a cube. The goal is to create the star schema depicted in Figure 10-25.

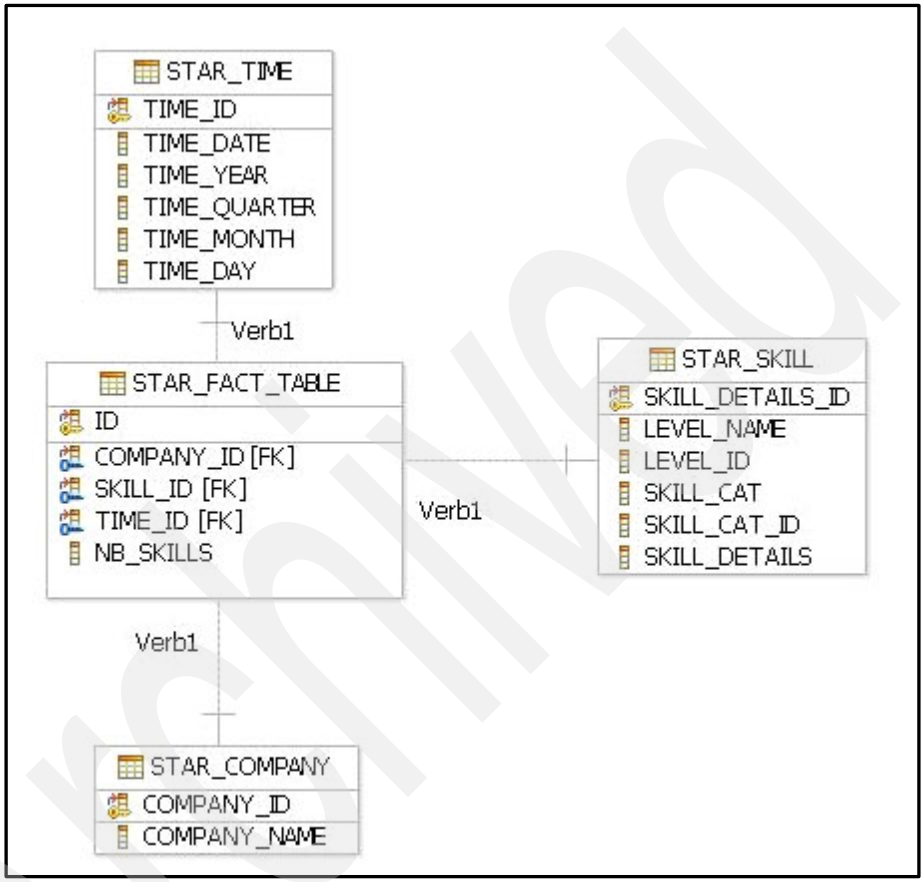


Figure 10-25 Star schema

A star schema is a database representation that is optimized for OLAP query. It is composed of a fact table and a number of dimension tables. The fact table, which is at the center of the schema, represents the measured values, while the dimensions tables, which form the star branches, represent the data descriptions.

The central table, the fact table, contains NB_SKILLS, the number of distinct skills per offer as a measure value. Of course, the others columns are foreign keys that refer to the three dimensions tables and an ID column. The SKILL and the TIME table contain several columns to allow hierarchies in the cube.

To create this schema, we use data flows to enable use of the sequence operator to generate ID columns. First, create a data model from the database by reverse engineering. Then expand the current project tree in the Data Project Explorer, right-click the **Databases** folder, and select **New** → **Physical Data Model**. On the successive pages of the wizard, select **Create from reverse engineering** and **Use an existing connection**. Check off the OLAPANL, TEMP, and TXTNAL schemas and click **Finish**. In the Data Project Explorer, the physical data model can be expanded and viewed.

Next, create the COMPANY dimension table. The COMPANY dimension table is the easiest one, because it contains only the name of the different companies and an ID number, which will later be used as the primary key. It is constructed as shown in Figure 10-26.

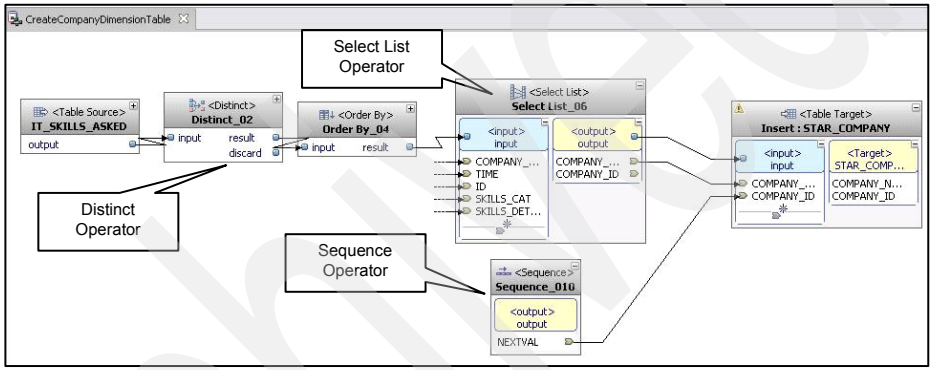


Figure 10-26 Flow for dimensions

In the Properties view of the Distinct operator, select COMPANY_NAME as the sorting value. COMPANY_NAME is also the sort key column in the Order By operator.

In the Select List operator, select COMPANY_NAME. A supplementary result column named COMPANY_ID (of type INTEGER) is added.

In the Sequence operator, select SEQ1, which is in the TEMP schema (this sequence operator uses sequence objects to generate numbers that can be used to fill an ID column, such as the auto-incremented command in SQL).

The target table is created by right-clicking the output port of the Select List operator and selecting **Create Suitable Table …**.

After expanding the Select List operator and the Target Table operator, the link between their COMPANY_ID columns is manually deleted. Instead, the Sequence operator's NEXTVAL column is connected to the target table's

COMPANY_ID column. The TIME dimension table is created in the same way as the COMPANY table. However, more columns are introduced to later allow a hierarchy in the OLAP cube. The result is depicted in Figure 10-27.

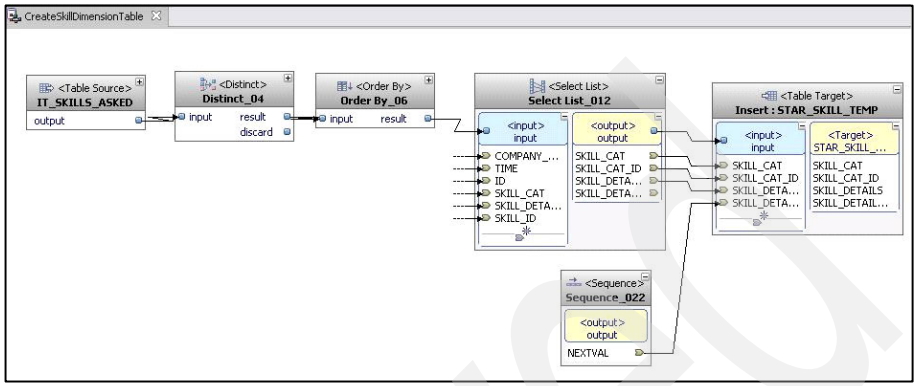


Figure 10-27 Resulting flow

Use SKILLS_DETAILS as the sorting key for the Distinct and the Order By operators. On the Select List operator, select SKILL_CAT, SKILL_ID (but rename it SKILL_CAT_ID), and SKILL_DETAILS as the result column. Then add one more column of INTEGER type and name it SKILL_ DETAILS_ID. Finally, create the STAR_SKILL_TEMP table in the TEMP schema.

In the second flow use the SKILL_HIERARCHY table exported from the taxonomy editor. For the Table Join, set SKILL_HIERARCHY.SUBLEVEL_NAME = STAR_SKILL_TEMP.SKILL_CAT as the condition and we set the result star model in Figure 10-28.

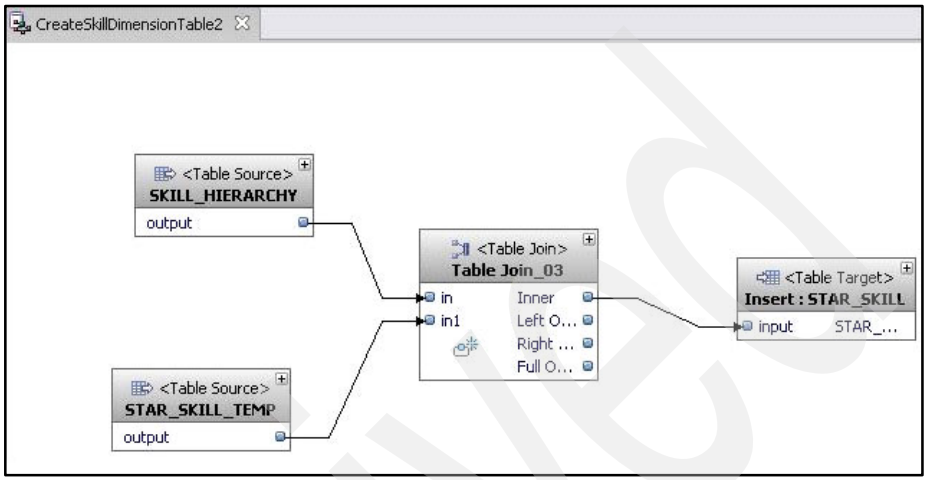


Figure 10-28 Result star model

The result column is shown in Figure 10-29.

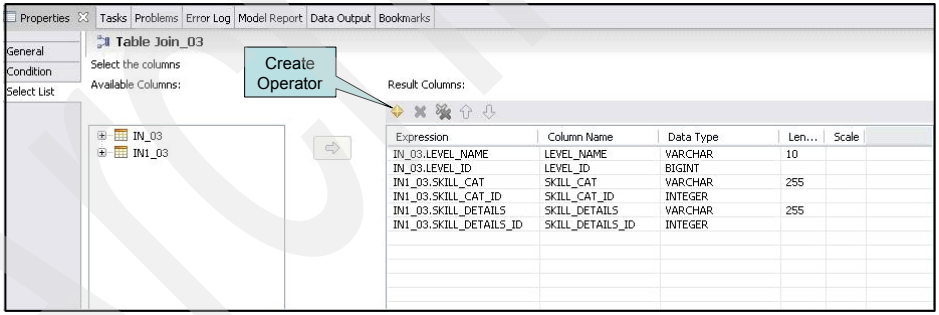


Figure 10-29 Result column

The fact table contains the measures and the keys to the dimension tables. There is only one measure, the numbers of distinct skills per jobs offering. To create this fact table, use a simple join between the different dimension tables, the IT_SKILLS_ASKED table, and the calculated fact value. It is depicted in Figure 10-30.

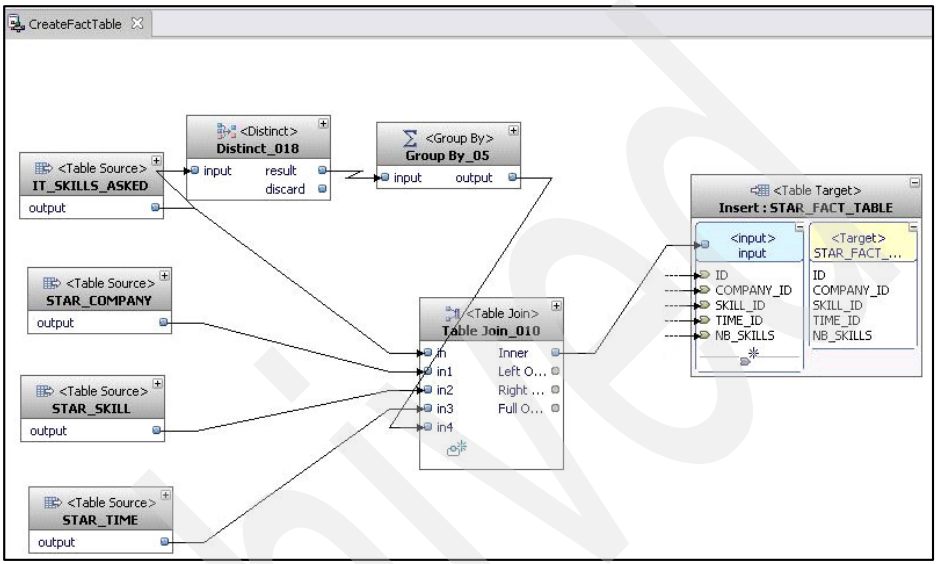


Figure 10-30 Fact table

In the Distinct operator, select SKILLS_CAT and SKILLS_ID as the sorting key. In the Group By operator, on the Select List tab, only select the ID column. Then add a new one by clicking the create operator (see Figure 10-29 on page 316), and define it with the expression `COUNT (`*`INPUT_XX`*`.SKILL_CAT)`, where INPUT_XX identifies the number of the Distinct operator. On the Group By tab, select the ID column.

In the Join operator, construct the following condition (with the correct INPUT number corresponding to the flow) using the SQL Expression® builder:

```
IN_010.COMPANY_NAME = IN1_010.COMPANY_NAME AND
IN2_010.SKILL_DETAILS = IN_010.SKILL_DETAILS AND
IN3_010.TIME_DATE =IN_010.TIME AND
IN_010.ID = IN4_010.ID
```

The Join operator Select List is depicted in Figure 10-31.

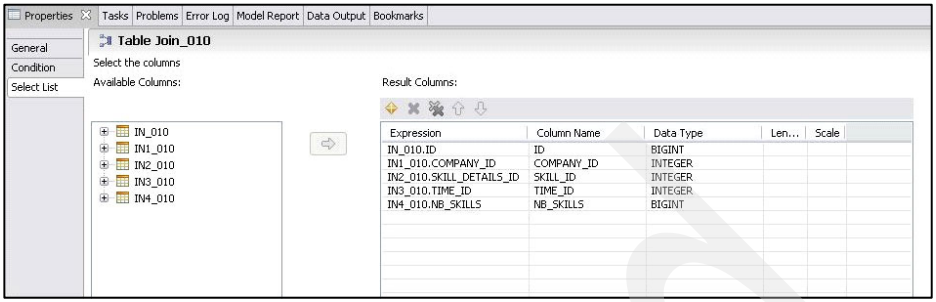


Figure 10-31 Join operator select list

Again, select **Create Suitable Table…** to create the target table STAR_FACT_TABLE in the OLAP schema.

In order to create an OLAP cube, define the integrity constraint on the tables, that is, to declare the primary and foreign keys.

- Expand the tree of the model until you can see the tables of the star schema.
- Click the **STAR_COMPANY** table. On the Columns tab of the Properties view, check the **COMPANY_ID** box as the Primary key.

Do the same with **TIME_ID** for the STAR_TIME table, **SKILLS_DETAILS_ID** for the STAR_SKILL table, and **ID** for the STAR_FACT_TABLE.

Then right-click the fact table and select **Add Data Object** → **Foreign Key**. Select **STAR_COMPANY**. A new relationship appears on the Data Project Editor view, and its Properties view opens. On the Details tab, on the Parent frame, select the correct unique constraint.

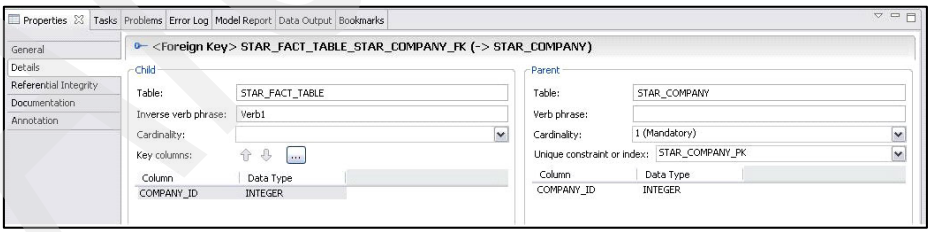


Figure 10-32 Properties view

Finally, this last step is repeated for the two others dimensions.

10.3.3 Creating an OLAP cube

Now that the star schema has been successfully created, we can now construct an OLAP cube based on the star schema and deploy it to the database. With the Cube View extension, DB2 is able to handle metadata and OLAP objects. We will use the Quick Advisor Wizard to define a cube model, then specify its parameters, and finally we will create a cube instance.

First, we create a new OLAP project by selecting **New** → **Data Design Project** and a Physical Data Model by right-clicking the **Data Model** folder in the Data Project Explorer. On the successive pages of the Physical Model creator wizard, select **Create from reverse engineering** → **Use an existing connection**.

Second, we create a cube model using the Quick Wizard Advisor. In the Data Project Explorer, we expand the **Data Model** folder, right-click the correct schema, and select **Add Data Object** → **Cube Model - Quick Start.** In the wizard, we specify **FACT_TABLE** as the fact table.

We now have to add levels and hierarchies to the different dimensions. In the Project Data Explorer, we expand the **TIME** dimension tree and right-click the **Level Folder** and select **Add Level**. In the Properties view, on the General tab, we name this level Calendar Day Level. On the Level key tab, we click the **Add level key attributes** button, and choose **TIME_DAY**, **TIME_MONTH**, **TIME_QUARTER**, and **TIME_YEAR**. On the Related tab, we click the **Add related attribute** button, and choose only **TIME_DAY**. We repeat the previous steps for **TIME_MONTH**, **TIME_QUARTER**, and **TIME_YEAR**. Then we right-click the **Hierarchy** folder and choose **Add Hierarchy**. In the Properties view, on the General tab, we name the hierarchy Time Hierarchy. On the Levels tab, we click the **Add level** button, click **Select All** and **OK**. We rearrange the order of the level with the arrows to obtain an ascending hierarchy from Calendar Year Level to Calendar Quarter Level, Calendar Month Level, and Calendar Day Level. Finally, we check the **All level** box and name it All Time.

For the SKILL and the COMPANY dimension, we follow a similar approach. For SKILL, we define the levels LEVEL_NAME, SKILL_CAT, and SKILL_DETAILS. For COMPANY, we define only one level COMPANY_NAME.

The OLAP model is depicted in Figure 10-33.

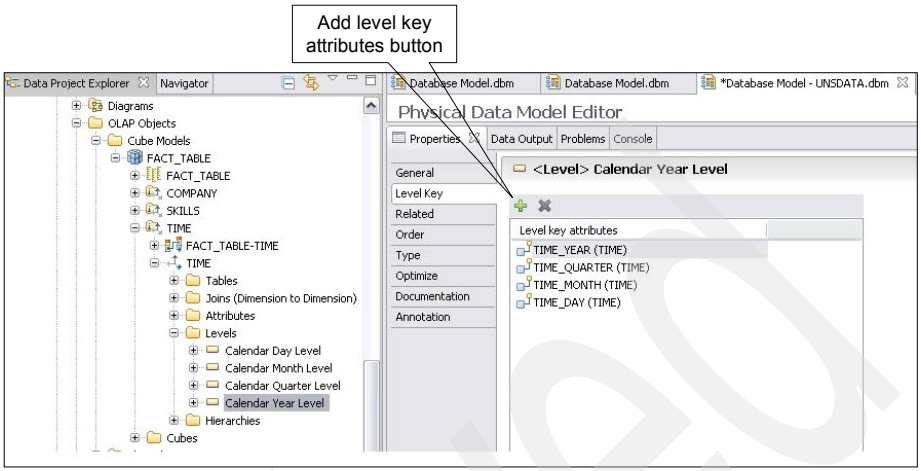


Figure 10-33 OLAP model

Now that we have defined a cube model, we will now create a cube from the cube model. In the Data Project Explorer, expand **Unstructured Tutorial - Data Model → Data Model → Database Model.dbm → [your schema] → OLAP Objects**, right-click the **Cubes** folder, and select **Add Cube**. Click the newly created cube. In the Properties view, on the General tab, type Jobs Analysis Cube in the Name field. On the Dimensions tab, click the **Add Cube Dimensions** button in the toolbar, then **Select All** and **OK**. Expand the **Jobs Analysis Cube**. On the Properties view of the Cube Facts, go to the Measures tab and click the **Add Measure** button. Select **NB_SKILLS** and click **OK**. For each dimension, verify that the hierarchy is selected (select the **Properties** view and then the **Hierarchy** tab).

The Properties view for the OLAP model is shown in Figure 10-34.

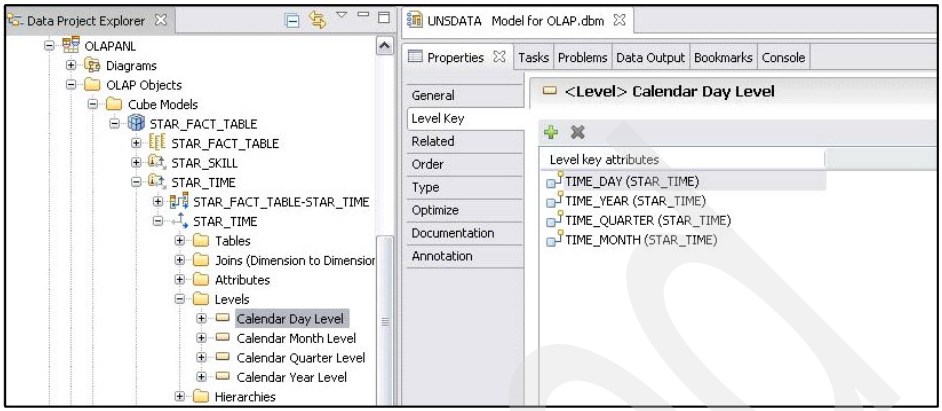


Figure 10-34 Properties view for OLAP model

Now validate the OLAP objects and deploy them to the database. In the Data Project Explorer, right-click the correct schema and select **Analyze Model**. Select the default parameters and click **OK**. In the Console view, check for errors and correct them. When there are no more errors, deploy the metadata to the database. To do this, right-click the **Jobs Analysis** cube model in the Data Project Explorer, and select **Deploy to the database**. The Deploy OLAP Objects wizard opens. Now specify the database as the target database, and click **Finish**. You may receive a message that prompts you to configure the database for OLAP objects. Click **Yes**. The database is then configured for OLAP objects and the OLAP objects are deployed.

10.4 Creating an Alphablox cube

In this section, we build an Alphablox Cube from the Cube View metadata. A DB2 Alphablox Cube stores pre-computed results in memory, not on disk. Any results not stored in memory remain in the underlying database. The cube retrieves results on an as-needed basis by sending SQL queries to the database. The results from the query are then stored in memory and are instantly accessible to DB2 Alphablox applications. This system allows a quick access into the data.

Alphablox cubes generate reports using the metadata and data available in relational databases. For accessing this underlying data, a DB2 Alphablox cube requires a data source definition. Log into the DB2 Alphablox Admin Pages as a user with administrator rights. From a Web browser, enter:

```
http://serverLocation:9080/AlphabloxAdmin/home
```

Then click the **Administration** tab, click the **Data Sources** link, and then click **Create**. Enter the following information and specify the correct values for Server Name, Default Username, and Default Password. Finally, click the **Save** button to save the data source. The result is depicted in Figure 10-35.

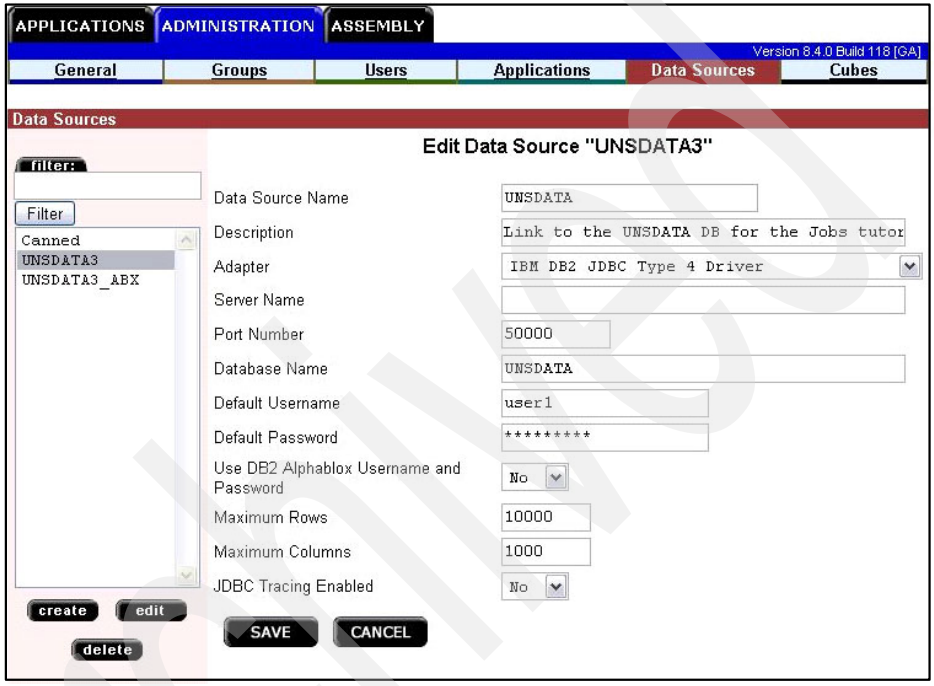


Figure 10-35 Data source

The DB2 Alphablox Cube Server needs its own data source to access any Alphablox Cube. Click the **Administration** tab, click the **Data Sources** link, and then click **Create**. Then enter the information depicted in Figure 10-36.

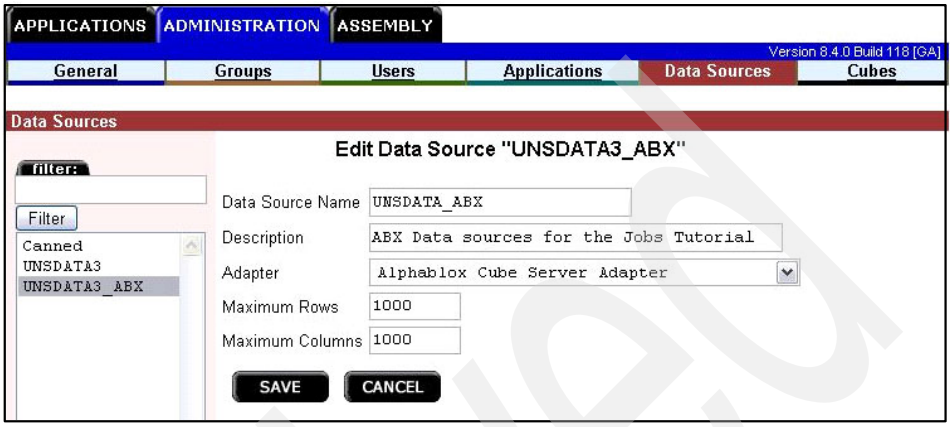


Figure 10-36 Data source information

Now define the Alphablox cube, using the predefined Cube Views metadata to quickly generate an Alphablox cube. On the Administration tab, click the **Cubes** link and then the **Create** button. In the DB2 Alphablox Cube Administration window, define the new cube. Then enable the Cube View settings and define the metadata to be used. Check the **Enable DB2 Cube Views Settings** box, select **OLAP.STAR_FACT_TABLE** in the Cube Mode menu and **Jobs Analysis Cube** for the Cube field. Then choose **Use Business Names**. Check the **Import cube definition on start, rebuild and edit** option. With this option, the Alphablox cube will always take the latest data present in the Cube View metadata on each start, build, or edit of the cube. Click **Import Cube Definition** and then **Show Log Import** to see the result, as depicted in Figure 10-37 on page 324.

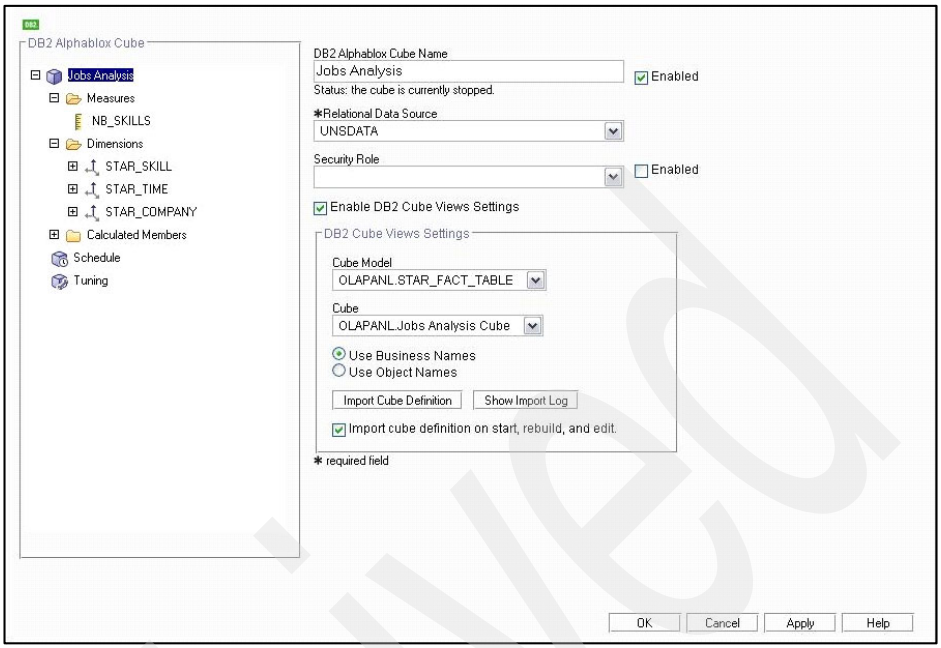


Figure 10-37 Alphablox cube

When the cube has been created, chose the **Enabled** option so that the cube will automatically start when the server is started. Here we start the cube manually using the Alphablox Admin Pages. Once again, click the **General** tab under the Runtime Management section and click the **DB2 Alphablox Cubes** link. Then select the **Jobs Analysis Cube** and click **Start**.

10.4.1 Developing an Alphablox application report

In this section, we use Alphablox to quickly and easily create an OLAP application, which can be visualized from any Web browser. Alphablox uses predefined Java components, called Blox, to retrieve data from the data source and to display it in the browser. The reports created use JSP™ tags to call and define the parameters of the Blox.

First, create and deploy an application on the WebSphere® server. Then use the DHTML Query Builder, a special feature from Alphablox, to construct and test the queries to the OLAP cube. Finally, employ these queries in the PresentBlox.jsp page, which will be used by the application to display the OLAP report.

Use the following symbols to represent the installation directories of DB2 Universal Database™, WebSphere Application Server, and DB2 Alphablox:

- <DB2_DRIVER_PATH>
- <WAS_HOME>
- <ABX_HOME>

First, use the Alphablox to generate an EAR file. Click **Applications** under the Administration tab and click **Create**. Enter JobsAnalysis as the Name and Display Name. Let the other fields keep their default values and click **Save**. Now connect to the WebSphere Administrative Console using the appropriate user name and password at the following location:

```
https://serverLocation:9043/ibm/console/logon.jsp
```

Select **Application** → **Install New Application**, and then select the correct path for the EAR file, which is typically <ABX_HOME>\installableApps\. On the next pages, leave the default options and just select our application when asked to. When finished, click **Enterprise Application** in the Applications tab and start the Jobs Analysis application, as depicted in Figure 10-38.

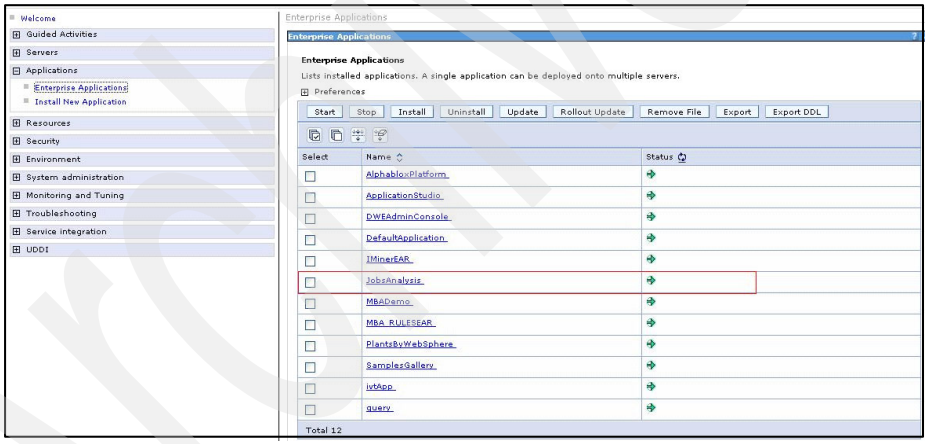


Figure 10-38 Applications list

When you open the Application tab of the Alphablox Admin Pages, you can now see the application running, as depicted in Figure 10-39.

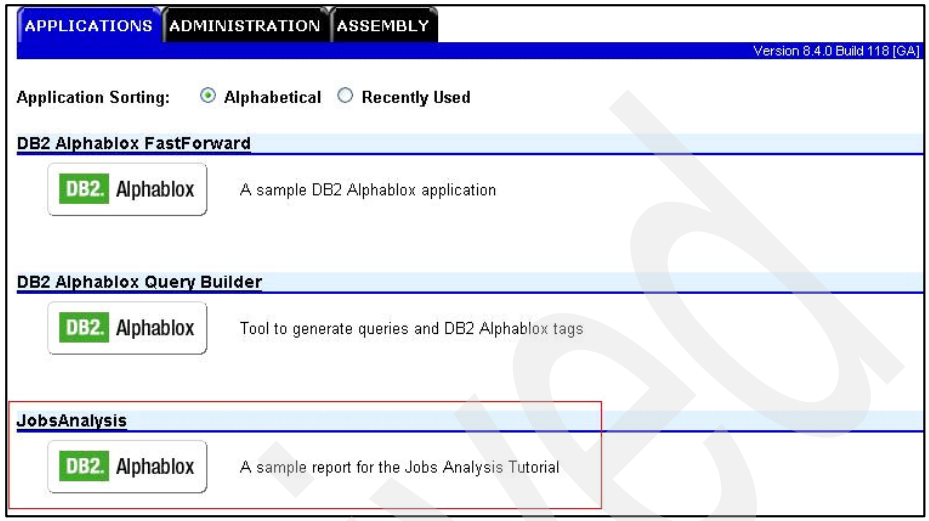


Figure 10-39 Alphablox application

So far, the application is empty and does nothing, so you need to define the report page using the JSP Alphablox tag and OLAP queries. To do that, use the DHTML Query Builder to generate and test the queries. The DHTML Query Builder is a test environment accessible from the Alphablox Admin Pages. You can test the queries to the cube and see the results in a grid and a chart as they could be showed in the report.

Log into the Alphablox Admin Pages, click the **Assembly** section, click the **Workbench** link and then the **Query Builder** link. On the Query Builder window, click the **Connection Settings** button. Select the correct data source and enter the correct user name and password to access the database. Then click **Connect**. On the Query frame, copy and paste the following statement:

```
SELECT DISTINCT( crossjoin ( {
[Jobs Analysis].[STAR_COMPANY]}, {[Jobs Analysis].[STAR_SKILL]})) ON
AXIS (0),
DISTINCT ({[Jobs Analysis].[STAR_TIME]}) ON AXIS (1)
FROM [Jobs Analysis]
```

This is a MQX statement that selects and displays the data from the Jobs Analysis Cube. The company and skill dimension are the columns of the grid and the time dimension is the rows. For more information about the MQX syntax, refer to the *DB2 Alphablox Cube Server Administrator's Guide*, SC18-9433.

Now click **Execute Query**. If there is no error, the results are displayed on the bottom of the page in the PresentBlox frame, as depicted in Figure 10-40. You can now double-click the different dimension to drill-down into the hierarchy's level. Note that the query changes each time you drill-down/up through the data.

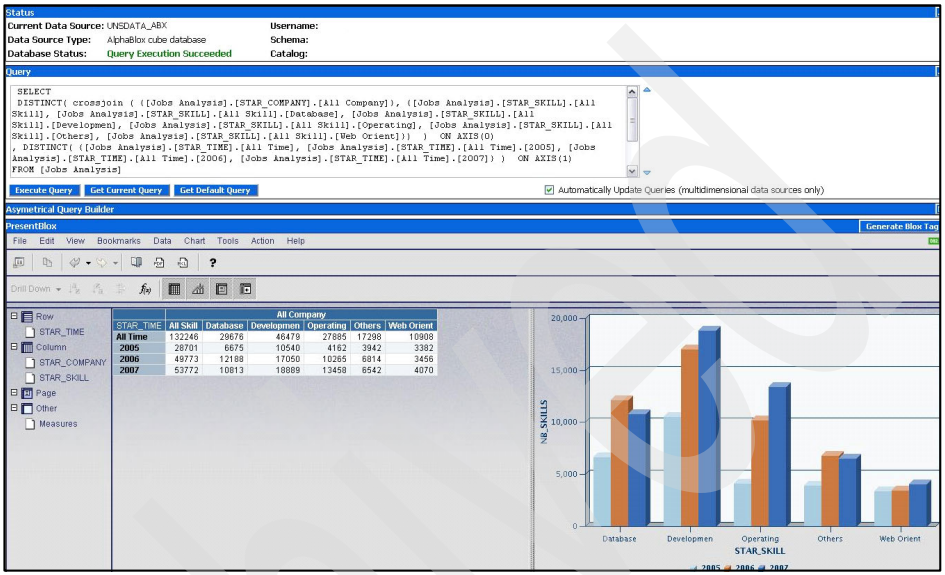


Figure 10-40 Query results

The Web application will use a PresentBlox.jsp page to construct the visual report for Web browsers. This page is a mix of HTML/CSS commands for the style, JSP tags for calling the Alphablox components, and MQX statements for querying the cube.

Open a text editor, and copy and paste the statements shown in Example 10-1 and save it under the name PresentBlox.jsp.

Example 10-1 PresentBlox.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
<HEAD>
<%@ taglib uri="bloxtld" prefix="blox" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>PresentBlox.jsp</TITLE>
<blox:header></blox:header>
</HEAD>
<BODY>
<P>Jobs Analysis : View of the Alphablox Cube</P>
<blox:present id="MyPresentBlox">
<blox:data dataSourceName="UNSDATA_ABX"
query="
SELECT DISTINCT( crossjoin ( {
[Jobs Analysis].[STAR_COMPANY]}, {[Jobs Analysis].[STAR_SKILL]})) ON AXIS (0),
DISTINCT ({[Jobs Analysis].[STAR_TIME]}) ON AXIS (1)
FROM [Jobs Analysis]
">
</blox:data>
<blox:chart/>
<blox:grid/>
</blox:present>
</BODY>
</HTML>
```

There are three important Blox used:

1. Data Blox: Defines the source name and query to be used.
2. Chart Blox: To display a chart.
3. Grid Blox: To display a grid.

Copy this JSP to the machine hosting the WebSphere Application Server:

```
<WAS_HOME>\AppServer\installedApps\<hostname>\JobsAnalysis.ear\JobsAnal
ysis.war\
```

You can now access the application and see the report at the following location:

```
http://serverLocation:9080/JobsAnalysis/PresentBlox.jsp
```

That application will appear as depicted in Figure 10-41.

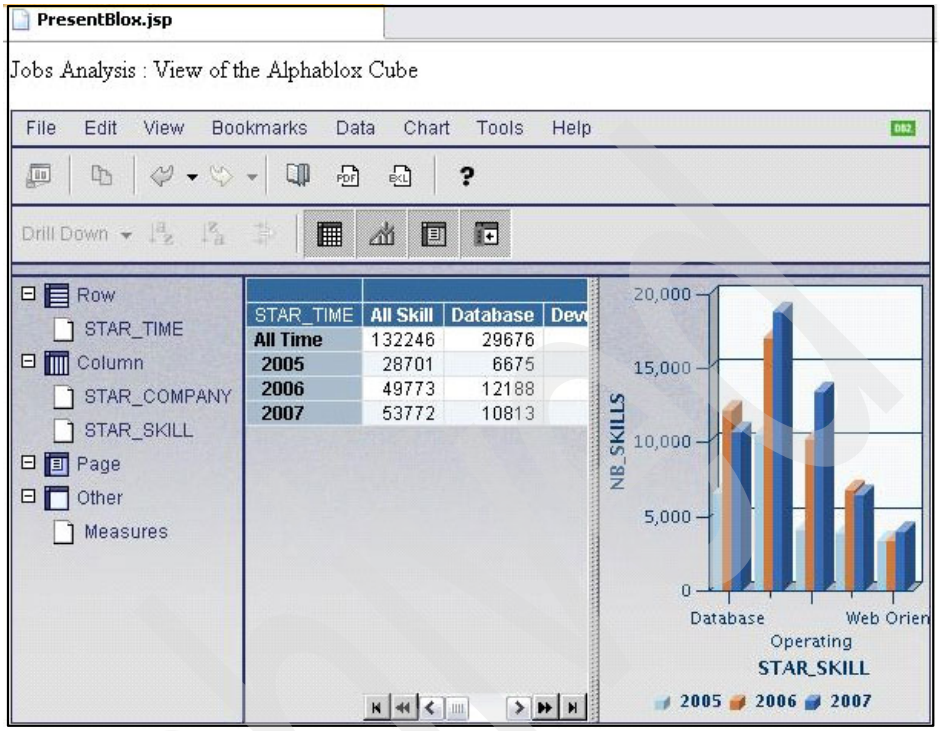


Figure 10-41 PresentBlox

Analyzing unstructured data can bring new and useful information to the decision makers in companies. Using the new mining tools in combination with the OLAP and reporting tools makes it simple to extract and make meaningful pieces of information available, which can be used in making business decisions.

A

DB2 Warehouse data mining algorithms: a deep dive

In this appendix, we provide a detailed discussion and description of the mining methods and algorithms provided in IBM DB2 Warehouse Version 9. We discuss modeling (creating models) and scoring (applying models) for 11 algorithms available in DB2 Warehouse for the data mining methods of clustering, classification, regression, associations, and sequences.

Clustering

A clustering algorithm tries to find clusters, groups of similar records, in the data it analyses. The records of one cluster should be homogeneous, and the records of two different clusters should be as heterogeneous as possible.

There is a large variety of clustering algorithms, which can be characterized by the:

- Result structure: hierarchical versus non-hierarchical (or partitioning), disjoint versus non-disjoint, exhaustive versus non-exhaustive
- The clustering process: iterative versus non-iterative, agglomerative (bottom up) versus divisive (top down)
- The similarity criterion
- The characterization of the resulting clusters: distribution based versus center based.

An overview on clustering techniques can be found in *Data Clustering: A Review*, by Jain, et al.

Demographic Clustering

The Demographic Clustering kernel implements a non-hierarchical, iterative, and distribution-based clustering algorithm whose similarity criterion is based on the so-called Condorcet criterion: The algorithm tries to find the partition with maximum Condorcet value. The number of clusters that best partitions the data is automatically calculated.

Algorithm

What is a similarity (or distance)? Similarity and distance are normalized numeric values between 0 and 1 that indicate how similar, or how far away from each other, two values are. The overall distance or similarity of two data records, each of them consisting of many data fields, is computed from a weighted sum of the distance or similarity values of each field.

What is a distribution based clustering algorithm? In distribution based clustering, a cluster is represented by a statistical distribution of the records it contains. For example, a cluster distribution could record the fact that the cluster contains four blue, eight green, and no red records. Four is called the frequency of the value blue in the distribution. If a record is to be added to this cluster, the distribution is used to calculate a similarity or distance between the new record and the cluster. In the example above, and with default 0/1 similarity, a blue record would have a similarity of 4/12 * 1 + 8/12 * 0 = 0.3333 with the cluster.

A field can be categorical, discrete numeric, or continuous. A categorical field contains non-numeric textual values, and the similarity and distance, between two of these values is calculated by default as follows: two equal values have a similarity of 1 and a distance of 0, two different values have a similarity of 0 and a distance of 1. The user can of course change this default. For example, if a taxonomy or hierarchy exists for the categorical values, the degree of relatedness can be used for a more fine-grained similarity measure. Following this approach, the two categorical values “university professor” and “high school teacher” of the field “profession” could be attributed a similarity of 0.7 because both of them fall into the profession subgroup “education” of the profession group “professions requiring academic education”, whereas the two values “university professor” and “motorcar mechanic” would be assigned the similarity 0. In “Parameters and advanced options” on page 335, we show how a similarity matrix can be defined within Demographic Clustering in order to express this fine-grained similarity measure. More generally, if the similarity matrix $M(x_1,x_2)$ assigns a similarity s value between 0 and 1 to each value pair (x_1,x_2), the similarity between a categorical value x and a cluster C described by N values x_j that appear with probabilities pj in the cluster is:

$$s(x,C) := \sum_{j=1..N} M(x,x_j)\, p_j$$

For a numeric field, Demographic Clustering uses a Gaussian similarity function with a fixed similarity scale, which means a fixed width of the Gaussian curve. For computing the similarity s between a field value x and a cluster C(μ) defined by the mean value μ of its value distribution, the formula is:

$$s(x,C) := \exp\left(-\ln 2 \, (|x-\mu| / \delta)^2 \right)$$

Here, δ is the so-called distance unit or similarity scale. In DB2 Warehouse, the default similarity scale δ_i of field i is $\delta_i = \sigma_i/2$, where σ_i is the empirical standard deviation of the attribute values of field i in the training data. The user can overwrite this default δ_i by a user-defined value using the advanced option DM_setFldSimScale(fieldname, value). The default distance function | | is the Euclidean distance, which can be overwritten by other distance functions such as squared Euclidean or Chebychev, all normalized to the maximum distance between two values observed in the training data. The constant “-ln 2” causes the value of s to be 0.5 if the distance |x-μ| equals δ.

If there is more than one data field, the overall similarity of a data record **x** := $(x_1,\ldots,x_n)$ with a cluster C is the weighted mean of the single field values' similarities:

$$s(\mathbf{x},C) := \sum_{i=1..n} w_i \, s(x_i,C) \,/ \sum_{i=1..n} w_i$$

In DB2 Warehouse, all w_i are equal to 1 per default, but the default can be overwritten using the advanced option DM_setFldWeight(fieldname, value).

Demographic Clustering uses an optimality criterion that is based on a vote counting system that was proposed by the French mathematician Condorcet (Jean Antoine Nicolas de Caritat, marquis de Condorcet, 1743-1794). Adapted to the clustering problem, the Condorcet method assumes that each pair of data records (i,j), $i \neq j$, acts as a voter who votes for (against) putting the two records in the same cluster if their similarity s(i,j) is larger (smaller) than a fixed threshold α.

The Condorcet criterion for clustering now states that the best partition P is the one that maximizes the difference between pro and contra votes:

$$\text{pro}(P) - \text{contra}(P) = \sum_{i,j \text{ in same cluster}} (s(i,j) - \alpha) - \sum_{i,j \text{ in diff.clusters}} (s(i,j) - \alpha) \rightarrow \max$$

It can be shown that this optimality criterion is equivalent to the following one, which is easier to calculate:

$$\text{Cond}(P) := \sum_{i,j \text{ in same cluster}} s(i,j) - \alpha \sum_{C_i \in P} |C_i| \, (|C_i| - 1) \rightarrow \max$$

Here, $\{C_i\}$ is the ensemble of all clusters that form the partition P, and $|C_i|$ is the number of records in cluster C_i.

In DB2 Warehouse, the default similarity threshold α is 0.5. It can be set to another value between 0 and 1 using advanced option DM_setSimThresh(value), or by means of the similarity threshold input field in the Mining Settings page of the Clusterer operator Properties view in the Design Studio.

The Demographic Clustering algorithm is an iterative method that first creates an initial partition P_0 and then iteratively constructs improved partitions P_1, P_2, … until convergence is reached or a user-defined stop criterion terminates the run.

The following algorithm creates the initial partition P_0:

```
P0 := {}
for all data records xq do {
  for all n existing clusters Ci∈P0 , i =1..n, do {
    compute P0(i) assuming Ci := Ci U {xq}
    compute Cond(P0(i))
  }
  compute P0(n+1) assuming {xq} forms new cluster Cn+1
  compute Cond(P0(n+1))
  P0 := that P0(i) which maximizes Cond(P0(i))
}
```

And this algorithm optimizes the initial partition P_0:

```
P := P_0
do {
  for all data records x_q do {
    remove {x_q} from its current cluster
    for all n existing clusters C_i ∈P,  i = 1...n, do {
      compute P^(i) assuming C_i := C_i U {x_q}
      compute Cond(P^(i))
    }
    compute P^(n+1) assuming {x_q} forms new cluster C_n+1
     compute Cond(P^(n+1))
    P := that P^(i) which maximizes Cond(P^(i))
  }
while Cond(P) has improved
```

Parameters and advanced options

The following advanced options are directly supported by the Mining Settings window of the Clusterer operator in the DesignStudio.

Maximum number of clusters

This option sets a hard upper limit for the number of clusters. The predefined value is 10. The clustering model generated by Demographic Clustering will never contain more clusters than specified in this option, but it might contain less clusters.

Typing a value x into the field Maximum number of clusters is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setMaxNumClus(x).

Similarity Threshold

This option overwrites the default value of 0.5 for the similarity threshold α. The allowed value range is between 0 and 1. The higher the value of a, the higher will be the average homogeneity of the generated clusters, and the higher also the number of clusters.

Typing a value x into the field Similarity Threshold is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setSimThresh(x).

Field usage types

In the Column Properties window of the Clusterer's Properties view, each available input data field has a default usage type of System determined. That means the clustering kernel is free to decide which fields it will use as active fields for the clustering, and which ones are not useful because they are either (almost) single-valued or (almost) key-like or highly correlated with another field.

You can overwrite this behavior for any data field if you click the value **System determined** in the table column Field Usage Type. A menu pops up where you can select either **active** or **supplementary**. Active fields are always used for defining the clusters. Supplementary fields are not used for defining the clusters, but their value distributions within each cluster are written into the clustering model.

Modifying the column Field Usage Type in the Column Properties window is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setFldUsageType('fldName',x). If the first argument fldName is missing, the command alters the default usage type for all fields: DM_setFldUsageType(x). Allowed values for x are 1 (active) or 2 (supplementary).

Field types

The Column Properties view of the Clusterer's Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields, whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one categorical, that means textual, value, and no ordering, such as x < y, is assumed between different values. In the resulting clustering model, the field's statistics is stored and visualized in the form of a categorical pie chart, and not as a numeric histogram.

Alternatively, the field type can be set to categorical by typing the following command into the Optional Parameters free text field: DM_setFldType('fldName',0).

The following advanced options do not have a directly corresponding input field in the Clusterer operator properties view. You have to type the API command into the general purpose Optional Parameters field on the Mining Settings window.

Figure A-1 shows this for the parameter Similarity Scale, together with the power option -buf.

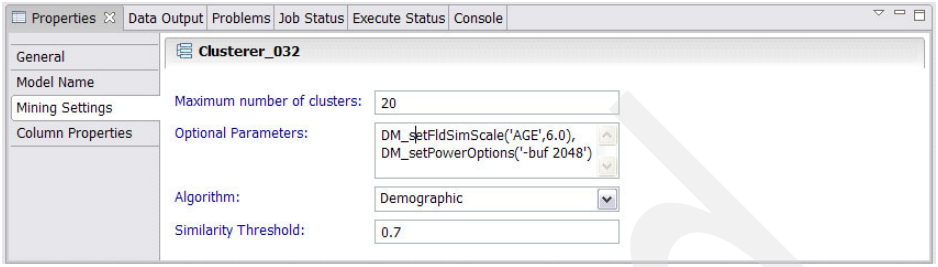


Figure A-1 Parameter settings on a similarity scale

Similarity Scale

Per default, DB2 Warehouse calculates similarities between two numeric field values by means of a Gaussian similarity function $s(x_1,x_2) := \exp(-\ln 2 (|x_1-x_2| / 0.5\sigma)^2)$, such that two values have a similarity of 1 if they are identical and of 0.5 if their difference is half the standard deviation α of the numeric field's overall value distribution. The command DM_setFldSimScale can be used to modify the width of this Gaussian similarity function. For example, if your data contains a field AGE, you can specify that two AGE values that differ by six years should have the similarity of 0.5 by typing DM_setFldSimScale('AGE',6) into the 'Optional Parameters' text window.

Similarity Matrix

Similarities between two categorical field values in Demographic Clustering are, per default, either 1 (if the two values are identical) or 0. Sometimes, a more fine-grained similarity measure is desirable. For example, if the field PROFESSION contains the values doctor, mechanic, nurse, professor, and teacher, we would like to somehow express that professor and teacher are quite similar because both of them are professions in the education sector, which require an academic education.

Regarding all professions, we might want to replace the default similarity matrix, which contains 1 on the diagonal and 0 for all other matrix cells, with the similarity matrix in Table A-1.

Table A-1 Similarity Matrix

	Doctor	Mechanic	Nurse	Professor	Teacher
Doctor	1	0	0.5	0.5	0.25
Mechanic	0	1	0.1	0	0
Nurse	0.5	0.1	1	0.5	0
Professor	0.25	0	0.5	1	0.7
Teacher	0.25	0	0	0.7	1

This can be done within DB2 Warehouse if we store the non-zero off-diagonal values in a three-column DB2 table, for example the in table PROF_SIM with the three columns VALUE1, VALUE2, and SIMILARITY, as depicted in Table A-2.

Table A-2 PROF_SIM

VALUE1	VALUE2	SIMILARITY
Doctor	Nurse	0.5
Doctor	Professor	0.5
Doctor	Teacher	0.25
Mechanic	Nurse	0.1
Nurse	Doctor	0.5
Professor	Teacher	0.7

Since the similarity matrix is symmetric, we only have to specify the triangle above the diagonal. Now we can specify this similarity matrix and activate it for the field PROFESSION by typing the following commands into the Optional Parameters text field of the Clusterer operator:

```
DM_addSimMat('ProfessionSimilarity','PROF_SIM','VALUE1',
   'VALUE2','SIMILARITY'),
DM_setFldSimMat('PROFESSION','ProfessionSimilarity')
```

Value weighting

Value weighting deals with the fact that particular values in a field might be more common than other values in that field. The coincidence of rare values in a field adds more to the overall similarity than the coincidence of frequent values. For example, most people do not have a Gold credit card. It is not very significant if two people do not have one, however, if they do, it is significant. Therefore, the coincidence of people not having a Gold credit card adds less to their overall similarity than the coincidence of people having one. You can use one of the following types of value weighting:

- *Probability weighting* assigns a weight to each value according to its probability in the input data. Rare values carry a large weight, while common values carry a small weight. This weight is used for both matching and non-matching records. Probability weighting uses a factor of 1/p, where p is the probability of a value.
- *Logarithmic weighting* assigns a weight to each value according to the logarithm of its probability in the input data. Rare values carry a large weight, while common values carry a small weight. This weight is used for both matching and non-matching records. Logarithmic weighting assigns a value of (-log(p)) to both the agreement information content value and the disagreement information content value. The number p is the probability of a value.

Each type of value weighting looks at the problem from a different angle. Depending on the value distribution, using one type or the other might lead to very different results.

Value weighting has the additional effect of emphasizing fields with many values because their values are less frequent than those of fields with fewer possible values. By default, the mining function does not compensate for this additional effect. You can select whether you want to compensate for the value weighting applied to each field. If you compensate for value weighting, the overall importance of the weighted field is equal to that of an unweighted field. This is so regardless of the number of possible values. Compensated weighting affects only the relative importance of coincidences within the set of possible values.

Value weighting within Demographic Clustering is activated for the field 'fieldname' by typing the following command into the Optional Parameters text window:

```
DM_setAlgorithm('Demographic','<ValWgt
field="fieldname">method</ValWgt>')
```

where *method* can take each of the four values prob, log, compProb, or compLog. The first two values stand for the uncompensated version of probabilistic

respectively logarithmic weighting, the last two values for the compensated version.

Field weighting

In the introductory text for Demographic Clustering, we have seen that a field weight factor w_i, which is 1 for all active fields i per default, enters into the similarity formula $s(x,C) := \sum_{i=1..n} w_i\, s(x_i,C) / \sum_{i=1..n} w_i$. The default value of 1 can be overwritten for certain fields by typing the following command into the Optional Parameters text box:

```
DM_setFldWeight('fieldname',x)
```

where x can be any positive real number.

Note: If one single field weight is set to a very large value, the clustering result becomes a bivariate data distribution result with respect to the one highly weighted field. The same effect can be attained if all but one field is marked as “supplementary” in the Column Properties page of the Clusterer properties.

Outlier treatment

For numeric fields, a valid value range can be defined, and once a valid range setting exists, an outlier handling method can be defined that specifies how to treat field values that lie below or above the allowed range. The following outlier treatment methods for values outside the valid range exist:

- 0: 'as is': The original value is kept unchanged. This method is the default.
- 1: 'as missing': The original value is replaced by SQL NULL.
- 2: 'as extreme': The original value is replaced by the upper, respectively lower boundary of the valid range if it lies above, respectively below the valid range.

And here are the two commands that DB2 Warehouse provides for this purpose:

```
DM_setFldOutlLim('fieldname',lower,upper)
DM_setFldOutlTreat('fieldname',methodID)
```

Here, lower and upper can be values of any integer, date or time, or floating point type; the type must fit with the corresponding field type. It is possible to specify only one range boundary; the other argument must then be 'CAST (NULL AS fieldType)'. methodID is either 0, 1, or 2.

Maximum execution time

Before starting a Demographic Clustering modeling run, the user can specify a desired maximum execution time by typing the following command into the Optional Parameters box:

```
DM_setExecTime(x)
```

where x is the desired maximum run time in minutes. When this option is specified, the clustering kernel tries to limit its runtime to not more than the desired amount, for example, by sampling the training data, or by terminating before the convergence criterion is reached. The kernel uses some heuristics based on CPU speed and other criteria for predicting the probable runtime. These heuristics are not always perfectly reliable, therefore it is possible that the actual runtime exceeds the desired one by a couple of minutes.

Number of bins

The command DM_setFldNumBins('fldName',x) states that the field *fldName* is to be discretized into x bins when collecting field value statistics.

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. *clause* contains the WHERE condition without the leading keyword WHERE. String constants within *clause* must be surrounded by two single quotes. For example:

```
DM_setWhereClause('GENDER=''m''').
```

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field 'fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

Demographic Clustering supports all power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the Clustering kernel.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.
- -IDM_MAX_DISCR_COUNT and its variants adjust the level of detail and the size of discrete value statistics contained in the clustering model.

Content of a Demographic Clustering model

Demographic Clustering models are stored as PMML <ClusteringModel modelClass="distributionBased" algorithmName="Demographic"> in the repository table IDMMX.CLUSTERMODELS. They contain the following information.

Model quality

Each Demographic Clustering model contains a model quality number Q in the range [0…1]. Q is the overall homogeneity of the clusters:

$Q := Avg_{x,y \text{ in same cluster}}\ s(x,y)$

Here, “Avg” means “average over all clusters and their records”, and s(x,y) is the similarity of the records *x* and *y*.

Field information for active fields

For each input data field that has been used as an active field in the clustering, the model contains the following measures:

- Value distribution: Frequencies of the discrete values for categorical fields, and frequencies of discretized histogram bins for numeric fields. The bin boundaries are automatically chosen by the clustering kernel in a suitable way, such that between about 15 and 80 bins with reasonably spaced and rounded boundary values are created.
- Field importance indicator: A value in the range [0…1], which indicates how significantly the field's value distribution in the various clusters differs from the value distribution in the overall population.

  More precisely, the formula for the importance I of field f is:

  $I(f) := N/(2N^2 - 2\sum_{C \in P} |C|^2) * \sum_{C \in P} \sum_{\text{values or bins } i} |n_{i,C}(f) - n_i(f) \bullet |C|/N|$

  Here, N is the total number of training data records, C is a cluster, P the partition consisting of all clusters {C}, and |C| is the number of records in cluster C. $n_{i,c}(f)$ is the number of records in cluster C having value i of field f. $n_i(f)$ is the total number of records having value i of field f, and therefore we would have $n_{i,c}(f) = n_i(f) \bullet |C|/N$ if the occurrence probability of value i of field f in cluster C was identical to its occurrence probability in the total population. The prefactor $N/(2N^2 - 2\sum_{C \in P} |C|^2)$ makes sure the result is in the range [0…1].
- Field homogeneity h: The mean similarity of training data records *x* and *y* with respect to the field f. h is a value between 0 and 1:

  $h(f) := Avg\ x \neq y\ s(x,y) = 2/(N(N-1)) \sum_{x \neq y} s(x,y)$
- Similarity Scale (for numeric fields): The value difference |x-y| to which a similarity of s(x,y) = 0.5 is assigned.

- Minimum value, maximum value, mean, standard deviation, sum, sum^2 (for numeric fields).

Field information for supplementary fields

For supplementary fields, only the value distribution is stored in the model.

Field-field correlations

The Demographic Clustering model contains field-field correlation coefficients for a subset of all possible field-field pairs (f,g), including the supplementary fields.

- If f and g are continuous numeric, DB2 Warehouse calculates linear correlation coefficients

  $corr(f,g) := \sum_{i=1...N}(f_i-\mu_f)(g_i-\mu_g) / N\sigma_f\sigma_g$,

  where (μ_f,σ_f) and (μ_g,σ_g) are mean and standard deviation of the values of fields f and g, and N is the number of data records.

- If f and g are discrete, that means discrete numeric or categorical; DB2 Warehouse calculates adjusted contingency coefficients:

  $cont(f,g) := \sqrt{(k/(k-1) \bullet X^2/(N+X^2))}$

  where X^2 is the result of the X^2 test that tests the hypothesis that the value of f significantly influences the value distribution of g (or vice versa). If f has k_f different values and g has k_g different values, then k is defined as $k := \min(k_f,k_g)$.

- No correlation is calculated between continuous and discrete fields.

- Furthermore, no correlations are calculated for fields that the mining kernel has removed from the list of active fields because they are either (almost) single valued or (almost) key-like.

Cluster information

The following defines the types of cluster information:

- Value distribution within the cluster for each field

- Homogeneity of each field's values within the cluster:

  $h(f) := Avg_{x\neq y\in C}s(x,y) = 2/(|C|(|C|-1)) \sum_{x\neq y\in C} s(x,y)$

- Chi square value χ^2 for each field f: The result of a χ^2-test that tests the hypothesis that the value distribution of f in the current cluster significantly differs from the field's value distribution in the total population. The difference is significant if the χ^2 value stored in the model is larger than the critical value $\chi^2_{crit}(k-1)$, where k is the number of different values or bins of field f. Tabular listings of $\chi^2_{crit}(k-1)$ for several confidence levels can be found in many books and tutorials on statistics.

- Minimum value, maximum value, mean, standard deviation, sum, sum^2 (for numeric fields).

Cluster-cluster similarities

For each pair of clusters C and D, C≠D, the mean similarity is:

$$\text{Avg}_{x \in C, y \in D}\ s(x,y)$$

Methods and operators

In this section, we describe the methods and operators for extracting model content into DB2 tables.

Cluster Extractor operator

The mining flow editor of the Design Studio contains an operator called Cluster Extractor that creates a table containing the following columns:

- ID (integer): ID of the cluster.
- NAME (varchar(128)): An automatically created name for the cluster. The name contains the combination of a field name and a field value that are most significant for this cluster (for example, 'no BankCard'). The field with most significant $\chi 2$ value within the cluster is selected for composing this name.
- SIZE (bigint): Number of records in the cluster.
- HOMOGENEITY (real): Homogeneity of the cluster; a value between 0 and 1.
- DESCRIPTION (varchar(1024)): A textual description of the most significant cluster properties, composed of single statements describing single field properties. For composing these field-related statements, DB2 Warehouse calculates how far the mean of the cluster μ_C is from the mean of the total population μ (in percentage of the standard deviation σ): $d(\mu_C,\mu) := (\mu_C - \mu) / \sigma$.

  - If $d(\mu_C,\mu) < -0.5$, the text 'low' is generated, for example 'INCOME is low'.
  - If $-0.5 = d(\mu_C,\mu) < 0.5$, the text 'medium' is generated.
  - If $d(\mu_C,\mu) = 0.5$, the text 'height' is generated.

  For categorical fields, for which no μ and σ exists, the text '… is predominantly value' is generated.

Quality Extractor operator

The Quality Extractor operator writes the clustering model's quality Q into a DB2 table. More precisely, the Quality Extractor, which can also be used with any other DB2 Warehouse mining model, writes four numbers: model quality Q and three detailed model quality numbers that are only available in classification and regression models: reliability, accuracy, and ranking quality. When applied to a clustering model, these columns in the output will contain SQL NULL.

Fields Extractor

The Fields Extractor operator writes a table containing the following information for each active mining field of the Demographic Clustering model:

- COLNAME: Name of the data column in the training data table.
- FIELDNAME: Per default equal to COLNAME; contains the field alias if an alias has been defined using the command DM_setFldAlias.
- MININGTYPE: 0 if the field is categorical, 1 if it is numeric.
- IMPORTANCE: A value in the range [0…1] that indicates how significantly the field's value distribution in the various clusters differs from the value distribution in the overall population. The exact formula is given in the previous section.

Correlations Extractor

The Correlations Extractor operator writes a table containing the following columns:

- FIELDNAME1: Name or alias name of an active or supplementary mining field
- FIELDNAME2: Name or alias name of a second active or supplementary field.
- CORRELATION: A linear correlation coefficient between -1 and 1 for continuous numeric fields, a contingency coefficient between 0 and 1 for discrete fields.

“Content of a Demographic Clustering model” on page 342 describes in more detail for which field pairs correlations are calculated.

SQL API commands and Custom SQL operator

The DB2 Warehouse mining SQL API offers some more model extractor methods for which no specific graphical mining flow operator exists. These commands can be used within a mining flow by means of the Custom SQL operator. The screen capture below shows how the API command DM_getClusters, which corresponds to the Cluster Extractor operator, is called within a Custom SQL operator. Figure A-2 shows the Properties view of a Custom SQL operator, which has been drawn to the DesignStudio' Mining Flow Editor canvas and whose input port has been connected to the table source 'CLUS_TABLE'.

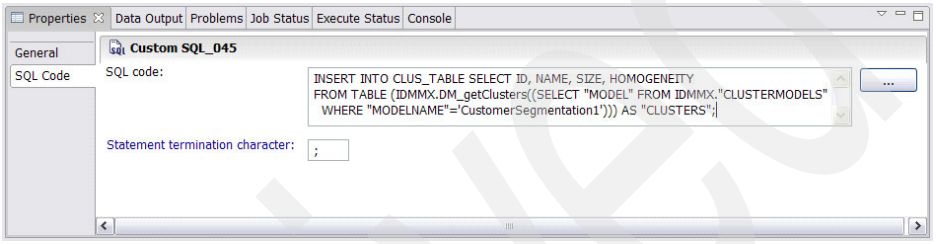


Figure A-2 Properties View of Custom SQL operator

This operator writes the cluster properties ID, NAME, SIZE, and HOMOGENEITY of each cluster contained in the model ‘CustomerSegmentation1’ into the table CLUS_TABLE.

The following model introspection commands for Demographic Clustering models, that means objects of type IDMMX.DM_ClusteringModel, are available:

- IDMMX.DM_expClusModel(clusModel): Returns the PMML model as a flat XML file in the form of a CLOB.
- IDMMX.DM_getClusMdlName(clusModel): Returns the model's name.
- IDMMX.DM_getClusMdlType(clusModel): Returns the type of the clustering model:'distributionBased'.
- IDMMX.DM_getMdlQuality(clusModel): Returns the model quality Q.
- IDMMX.DM_getNumClusters(clusModel): Returns the number of clusters contained in the model.
- IDMMX.DM_getClusterName(clusModel,position): Returns the name of the cluster at position 'position' in the model. Position is an integer between 1 and the number of clusters in the model.
- IDMMX.DM_getClusters(clusModel): This command is the API equivalent of the Cluster Extractor operator. It returns a 5-column table (ID, NAME, SIZE, HOMOGENEITY, and DESCRIPTION) in which each row describes one cluster contained in the model.

- IDMMX.DM_getClusterStats(clusModel): Returns a 7-column table (CLUSTERID, FIELDNAME, CATVALUE, LOWERBOUND, UPPERBOUND, FREQUENCY, and EXPECTEDFREQ) that contains for each cluster, each active field, and each value or value range (bucket) of this field, the actual number of occurrences in the cluster and the expected number of occurrences in the cluster, assuming that the relative distribution of field values in the cluster is identical to the entire population.

  If the field is a categorical field, the columns LOWERBOUND and UPPERBOUND contain SQL NULL. If the field is a discrete numeric field, the columns CATVALUE and UPPERBOUND contain SQL NULL. If the field is a discretized continuous field, the column CATVALUE contains SQL NULL.
- IDMMX.DM_getCorrelations(clusModel): Returns a three-column table (FIELDNAME1, FIELDNAME2, CORRELATION) which lists all field-field correlation numbers contained in the model.

Applications of Demographic Clustering

Demographic Clustering is a robust standard clustering method that can be applied to data and use cases in almost any industry, for example:

- Cross-marketing
- Cross-selling
- Customizing marketing plans for different customer types
- Deciding which media approach to use
- Understanding shopping goals
- Store profiling
- Deviation or fraud detection

The method can work both with predominantly categorical and with predominantly numeric data. However, for purely floating point data, runtimes might be considerably longer than with the Kohonen Clustering method.

Unlike Kohonen Clustering, Demographic Clustering is well suited for detecting small clusters, for example in deviation detection scenarios. Another advantage is that the optimum number of clusters is determined automatically by the algorithm and needs not be specified by the user.

Further reading

- “Techniques of Cluster Algorithms in Data Mining” by Rudolf Grabmeier and Andreas Rudolph in Data Mining and Knowledge Discovery, June 4, 2002
- Jain, et al, *Data Clustering: A Review*, ACM Comp. Surv., 31, 264-323

- An easy-to-read introduction to chi square tests can be found at:

  http://www.georgetown.edu/faculty/ballc/webtools/web_chi_tut.html
- Introduction to contingency tables:

  http://en.wikipedia.org/wiki/Contingency_table

Kohonen Clustering

The Kohonen Clustering algorithm is a neural network algorithm that interprets the available data fields as input neurons and the clusters to be detected as output neurons. Each input neuron is connected to each output neuron; the weights of these connections are modified and adapted during the learning phase. There are no intermediate neurons, therefore the Kohonen network can be characterized as single layer feedforward network. In the scientific literature, neural networks of this type are called Kohonen networks or Self Organizing Maps (SOM®). The output neurons are organized in a spatial structure called a *feature map*.

In Kohonen Clustering, the map is a two-dimensional square lattice, and each output neuron's position on this lattice can be described by a two-dimensional position vector containing non-negative integer coordinates. This is depicted in Figure A-3.

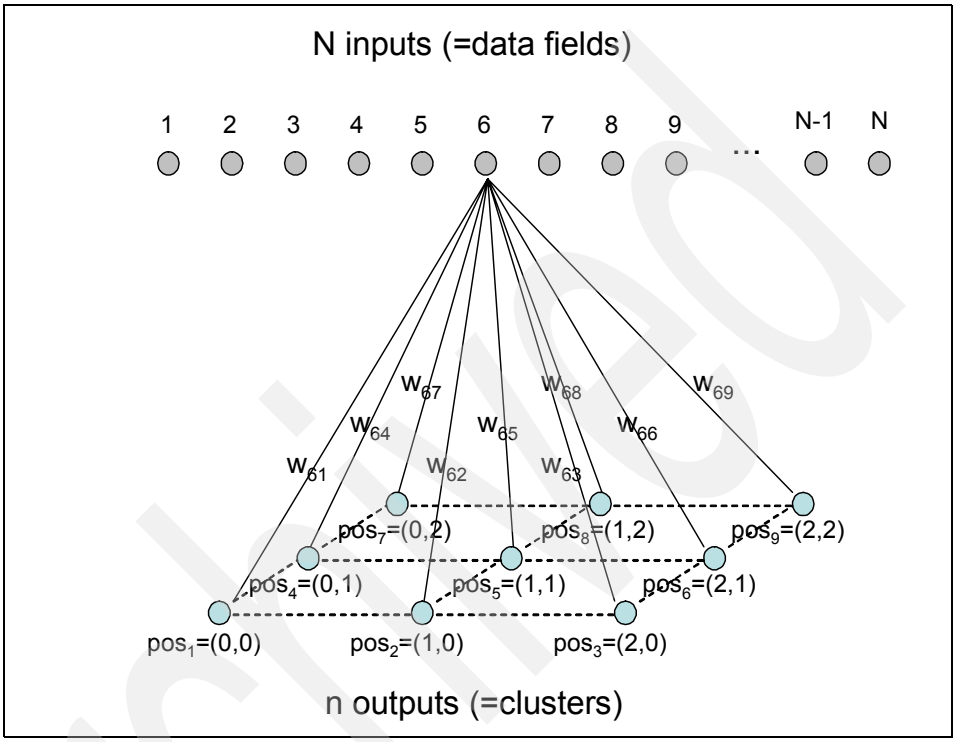


Figure A-3 Network Map

Algorithm

Neural network algorithms internally work with values between 0 and 1. Therefore, the training data stored in input tables must be normalized before it is fed into the neural network. For numeric fields, normalization means mapping the smallest value found in the training data to 0, and the largest one to 1. Categorical fields with m different values are mapped to m different binary fields with values 0 or 1. If the original categorical field contains the i-th value, the i-th corresponding binary field contains the value 1, and the m-1 other binary fields contain 0.

Each output neuron o_i carries a weight vector $w_i = (w_{1i},\ldots,w_{Ni})$, which can be interpreted as the N-dimensional coordinate vector of a cluster center. The Kohonen feature map tries to put the cluster centers at places that minimize the overall distance between all training records and the cluster centers. During the learning process, the data records {x} are presented one by one to the Kohonen network. Let BM be the index of the best matching output neuron for the current record x(t), which means $|x(t) - o_{BM}| \rightarrow \min$. Then all cluster centers w_i are adjusted towards x(t), the strength of the adjustment being the lower it is, the farther away an output neuron's position pos_i is from the best matching neuron's position pos_{BM}:

$w_i(t+1) = w_i(t) + \theta(|pos_i - pos_{BM}|, t)\ \alpha(t)\ (x(t) - w_i(t))$.

Here, T(d,t) is a strength function that monotonically decreases with increasing first argument d and that becomes more and more narrow with increasing time argument t. For example, T(d,t) could be a Gaussian T0 exp(-d2/2s(t)2). The so-called learning rate a(t), with 0 < a(t) < 1, is also decreased towards 0 with increasing time t.

The following pseudo code summarizes the Kohonen clustering algorithm:

```
Initialize weights wij(0) with random numbers
do {
  for all input records t do {
    fetch current record's data x(t)
    BM := output neuron with weight w_i(t) closest to x(t)
    update weights: w_i(t+1) = w_i(t) + θ(|pos_i - pos_BM|, t) α(t) (x(t) - w_i(t))
  }
  shrink θ(d, t), decrease α(t)
} while (not converged and not maxNumPasses reached)
```

The separability of clusters is not taken into account by this algorithm. You must specify a total number of passes. With each pass, the center vectors are adjusted to minimize the total distance between cluster centers and records. Also, the amount by which the vectors are adjusted is decreased. In the first pass, the adjustments are rough. In the final pass, the amount by which the centers are adjusted is rather small. Only minor adjustments are done.

Using the DB2 Warehouse mining API, you can select Kohonen Clustering by using the following command: DM_setAlgorithm('Kohonen'). In the DesignStudio, select **Kohonen** from the Algorithm drop-down menu in the Clusterer operator Properties view.

Parameters and advanced options

The following advanced options for Kohonen clustering are directly supported by the Mining Settings window of the Clusterer operator in the DB2 Warehouse DesignStudio.

Maximum number of clusters

This option sets a hard upper limit for the number of clusters. The predefined value is 9, corresponding to a Kohonen map layout of 3 x 3. In most cases, the clustering model generated by Kohonen Clustering will exactly contain the number of clusters specified as the maximum. In some cases, however, one or more of the initial clusters are dropped during the training process because they have lost all their records.

Typing a value x into the field Maximum number of clusters is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setMaxNumClus(x).

Note: If you specify a maximum number of clusters, you cannot specify a layout for the Kohonen feature map.

Number of passes

Per default, Kohonen Clustering performs 10 passes over the training data to adjust the neural network and to train the clustering model. You can manually increase or decrease that value. More than 10 passes can sometimes increase the quality of the resulting model at the cost of a longer runtime of the training process.

Typing a value x into the field Number of Passes is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setAlgorithm('Kohonen','<NumPasses>x</NumPasses>').

Note: The learning rate and the neighborhood function are calculated to match the number of passes. Therefore, you cannot specify these parameters.

Field Usage Types

In the Column Properties window of the Clusterer Properties view, each available input data field has a default usage type System determined. That means the clustering kernel is free to decide which fields it will use as active fields for the clustering, and which ones are not useful because they are either (almost) single-valued or (almost) key-like or highly correlated with another field.

You can overwrite this behavior for any data field if you click the value **System determined** in table column Field Usage Type. A menu pops up where you can select either **active** or **supplementary**. Active fields are always used for defining the clusters. Supplementary fields are not used for defining the clusters, but their value distributions within each cluster are written into the clustering model.

Modifying the column Field Usage Type in the Column Properties window is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setFldUsageType('fldName',x). If the first argument 'fldName' is missing, the command alters the default usage type for all fields: DM_setFldUsageType(x). Allowed values for x are 1 (active) or 2 (supplementary).

Field Types

The Column Properties view of the Clusterer Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields, whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one categorical, which means textual, value, and no ordering such as 'x < y' is assumed between different values. In the resulting clustering model, the field's statistics is stored and visualized in the form of a categorical pie chart, and not as a numeric histogram.

Alternatively, the field type can be set to categorical by typing the following command into the Optional Parameters free text field: DM_setFldType('fldName',0).

The following advanced options do not have a directly corresponding input field in the Clusterer operator properties view. You have to type the API command into the general purpose Optional Parameters field on the Mining Settings window. Figure A-4 shows this for the parameter Outlier Limits, together with the power option -IDM_MAX_DISCR_COUNT.

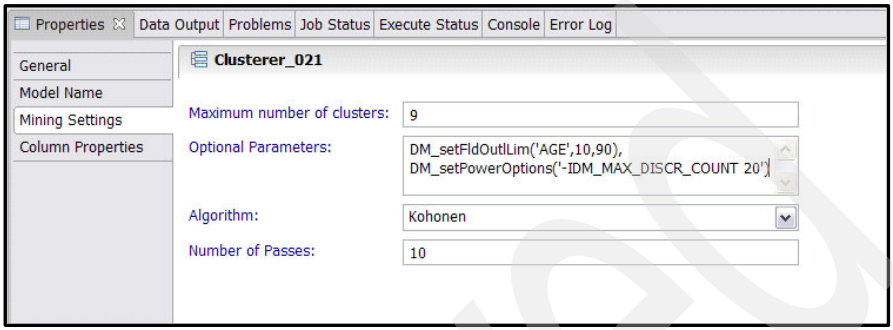


Figure A-4 Clusterer Operator Properties

Kohonen feature map layout

You can specify how the Kohonen feature map should look like by specifying a map layout. To define the number of rows x and the number of columns y of a Kohonen feature map, you can use the method DM_setAlgorithm in the Optional Parameters text box:

```
DM_setAlgorithm('Kohonen','<LayoutNumRows>x</LayoutNumRows>
  <LayoutNumColumns>y</LayoutNumColumns>')
```

In the command above, the number of clusters to be used during training is implicitly determined by specifying x rows multiplied by y columns = x y clusters. Therefore, you cannot explicitly specify the maximum number of clusters when you specify the map layout.

Outlier treatment

For numeric fields, a valid value range can be defined, and once a valid range setting exists, an outlier handling method can be defined that specifies how to treat field values that lie below or above the allowed range. The following outlier treatment methods for values outside the valid range exist:

- 0: 'as is': The original value is kept unchanged. This method is the default.
- 1: 'as missing': The original value is replaced by SQL NULL.
- 2: 'as extreme': The original value is replaced by the upper, respectively lower boundary of the valid range if it lies above, respectively below the valid range.

Here are the two commands that DB2 Warehouse provides for this purpose:

```
DM_setFldOutlLim('fieldname',lower,upper)
DM_setFldOutlTreat('fieldname',methodID).
```

Here, lower and upper can be values of any integer, date or time, or floating point type; the type must fit with the corresponding field type. It is possible to specify only one range boundary; the other argument must then be 'CAST (NULL AS fieldType)'. methodID is either 0, 1, or 2.

Number of bins

The command DM_setFldNumBins('fldName',x) states that the field 'fldName' is to be discretized into x bins when collecting field value statistics.

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within 'clause' must be surrounded by two single quotes. Example:

```
DM_setWhereClause('GENDER=''m''').
```

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field 'fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

Kohonen Clustering supports the following power options, which are described in detail in Appendix B, “Power options” on page 503:

- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.
- -IDM_MAX_DISCR_COUNT and its variants adjust the level of detail and the size of discrete value statistics contained in the clustering model.

Note: DB2 Warehouse Kohonen Clustering does not need much memory, so the power option -buf is of no use here.

Content of a Kohonen Clustering Model

DB2 Warehouse Kohonen clustering models are stored as PMML <ClusteringModel modelClass="centerBased" algorithmName="Kohonen"> in the table IDMMX.CLUSTERMODELS. They contain the following information:

Model quality

Each DB2 Warehouse Kohonen Clustering model contains a model quality number Q in the range [0…1]. Q is the overall homogeneity of the clusters as measured by the mean distance between the training data records x(t) and their best matching cluster centers $w_{BM}(t)$:

$Q := 1 - Avg_t |x(t) - w_{BM}(t)| / \sqrt{N}$.

Here, 'Avg' means average over all input data records and N is the number of normalized input fields.

Field information for active fields

For each input data field that has been used as an active field in the clustering, the model contains the following measures:

- Value distribution: Frequencies of the discrete values for categorical fields; frequencies of discretized histogram bins for numeric fields. The bin boundaries are automatically chosen by the clustering kernel in a suitable way, such that between about 15 and 80 bins with reasonably spaced and rounded boundary values are created.
- Field homogeneity h: The mean similarity of training data records x and y with respect to the field f. h is a value between 0 and 1:

  $h(f) := 1- Avg_{x \neq y}|f(x)-f(y)|$
- Minimum value, maximum value, mean, standard deviation, sum, sum^2 (for numeric fields).

Field information for supplementary fields

For supplementary fields, only the value distribution is stored in the model.

Field-field correlations

The Kohonen Clustering model contains field-field correlation coefficients for a subset of all possible field-field pairs (f,g), including the supplementary fields.

- If f and g are continuous numeric, DB2 Warehouse calculates linear correlation coefficients:

  $corr(f,g) := \sum_{i=1...N}(f_i-\mu_f)(g_i-\mu_g) / N\sigma_f\sigma_g$

  where (μ_f,σ_f) and (μ_g,σ_g) are mean and standard deviation of the values of fields f and g, and N is the number of data records.

- If f and g are discrete, that means discrete numeric or categorical, DB2 Warehouse calculates adjusted contingency coefficients:

  $cont(f,g) := \sqrt{(k/(k-1) \bullet \chi^2 / (N+ \chi^2))}$

  where χ^2 is the result of the χ^2 test that tests the hypothesis that the value of f significantly influences the value distribution of g (or vice versa). If f has k_f different values and g has k_g different values, then k is defined as $k := \min(k_f,k_g)$.
- No correlation is calculated between continuous and discrete fields.
- Furthermore, no correlations are calculated for fields that the mining kernel has removed from the list of active fields because they are either (almost) single valued or (almost) key-like.

Cluster information

- Value distribution within the cluster for each field.
- Homogeneity of each field f's values within the cluster:

  $h(f) := 1- Avg_{x \neq y \in C} |f(x)-f(y)|$
- Chi square value χ^2 for each field f: The result of a χ^2-test that tests the hypothesis that the value distribution of f in the current cluster significantly differs from the field's value distribution in the total population. The difference is significant if the χ^2 value stored in the model is larger than the critical value $\chi^2_{crit}(k-1)$, where k is the number of different values or bins of field f. Tabular listings of $\chi^2_{crit}(k-1)$ for several confidence levels can be found in many books and tutorials on statistics.
- Minimum value, maximum value, mean, standard deviation, sum, sum^2 (for numeric fields).

Cluster-cluster similarities

For each pair of clusters C and D, C≠D, the mean similarity $1 - Avg_{x \in C, y \in D} |x-y| / \sqrt{N}$, where N is the number of normalized input fields.

Methods and operators

In this section, we describe the methods and operators for extracting model content into DB2 tables.

Cluster Extractor operator

The mining flow editor of the Design Studio contains an operator called ‘Cluster Extractor’ that creates a table containing the following columns:

- ID (integer): ID of the cluster.
- NAME (varchar(128)): An automatically created name for the cluster. The name contains the combination of a field name and a field value that are most significant for this cluster (for example, 'no BankCard'). The field with most significant χ^2 value within the cluster is selected for composing this name.
- SIZE (bigint): Number of records in the cluster.
- HOMOGENEITY (real): Homogeneity of the cluster; a value between 0 and 1.
- DESCRIPTION (varchar(1024)): A textual description of the most significant cluster properties, composed of single statements describing single field properties. For composing these field-related statements, DB2 Warehouse calculates how far the mean of the cluster μ_C is from the mean of the total population μ (in percentage of the standard deviation σ): $d(\mu_C,\mu) := (\mu_C - \mu) / \sigma$.
  - If $d(\mu_C,\mu) < -0.5$, the text 'low' is generated, for example 'INCOME is low'.
  - If $-0.5 = d(\mu_C,\mu) < 0.5$, the text 'medium' is generated.
  - If $d(\mu_C,\mu) = 0.5$, the text 'height' is generated.

  For categorical fields, for which no μ and s exists, the text '… is predominantly value' is generated.

Quality Extractor operator

The Quality Extractor operator writes the clustering model's quality Q into a DB2 table.

More precisely, the Quality Extractor, which can also be used with any other DB2 Warehouse mining model, writes four numbers: model quality Q and three detailed model quality numbers that are only available in classification and regression models: reliability, accuracy, and ranking quality. When applied to a clustering model, these columns in the output will contain SQL NULL.

Fields Extractor

The Fields Extractor operator writes a table containing the following information for each active mining field of the Demographic Clustering model:

- COLNAME: Name of the data column in the training data table.
- FIELDNAME: By default equal to COLNAME; contains the field alias if an alias has been defined using the command DM_setFldAlias.
- MININGTYPE: 0 if the field is categorical, 1 if it is numeric.

- IMPORTANCE: A value in the range [0…1] that indicates how significantly the field's value distribution in the various clusters differs from the value distribution in the overall population. The exact formula is given in the previous section.

Correlations Extractor

The Correlations Extractor operator writes a table containing the following columns:

- FIELDNAME1: Name or alias name of an active or supplementary mining field.
- FIELDNAME2: Name or alias name of a second active or supplementary field.
- CORRELATION: A linear correlation coefficient between -1 and 1 for continuous numeric fields, a contingency coefficient between 0 and 1 for discrete fields.

“Content of a Demographic Clustering model” on page 342 describes in more detail for which field pairs correlations are calculated.

SQL API commands and Custom SQL operator

The DB2 Warehouse mining SQL API offers some more model extractor methods for which no specific graphical mining flow operator exists. These commands can be used within a mining flow by means of the Custom SQL operator. Figure A-5 shows how the API command DM_getClusters, which corresponds to the Cluster Extractor operator, is called within a Custom SQL operator. It shows the Properties view of a Custom SQL operator, which has been drawn to the DesignStudio's Mining Flow Editor canvas and whose input port has been connected to the table source 'CLUS_TABLE'.

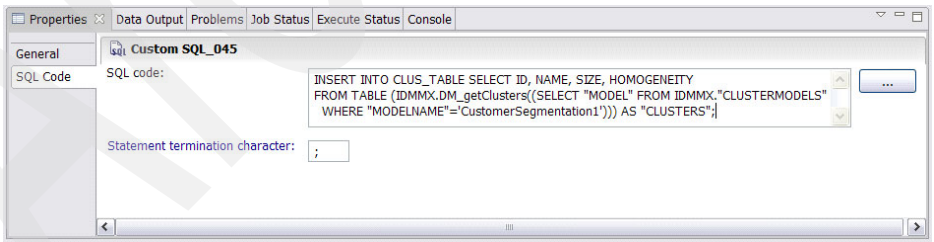


Figure A-5 Cluster Extractor Operator

This operator writes the cluster properties ID, NAME, SIZE, and HOMOGENEITY of each cluster contained in the model 'CustomerSegmentation1' into the table CLUS_TABLE.

The following model introspection commands for Kohonen clustering models, which means objects of type IDMMX.DM_ClusteringModel, are available:

- IDMMX.DM_expClusModel(clusModel)

  Returns the PMML model as a flat XML file in the form of a CLOB.

- IDMMX.DM_getClusMdlName(clusModel)

  Returns the model's name.

- IDMMX.DM_getClusMdlType(clusModel)

  Returns the type of the clustering model:'distributionBased'.

- IDMMX.DM_getMdlQuality(clusModel)

  Returns the model quality Q.

- IDMMX.DM_getNumClusters(clusModel)

  Returns the number of clusters contained in the model.

- IDMMX.DM_getClusterName(clusModel,position)

  Returns the name of the cluster at position 'position' in the model. Position is an integer between 1 and the number of clusters in the model.

- IDMMX.DM_getClusters(clusModel)

  This command is the API equivalent of the Cluster Extractor operator. It returns a 5-column table (ID, NAME, SIZE, HOMOGENEITY, and DESCRIPTION) in which each row describes one cluster contained in the model.

- IDMMX.DM_getFields(clusModel)

  Returns a four-column table (COLNAME, FIELDNAME, MININGTYPE, and IMPORTANCE) that contains, for each active field in the model, the name of the data column to which it refers, its name, and its mining type (categorical or numeric). For Kohonen clustering models, importance values are not available (SQL NULL).

- IDMMX.DM_getCorrelations(clusModel)

  Returns a three-column table (FIELDNAME1, FIELDNAME2, CORRELATION) that lists all field-field correlation numbers contained in the model.

Applications

Categorical input variables with lots of different values can slow down Kohonen Clustering mining considerably because in this situation the number of normalized 0/1 fields is much larger than the number of original input fields. Therefore, Kohonen Clustering is best suited for data with predominantly numeric data. Compared to Demographic Clustering, Kohonen clustering tends

to produce clusters of similar sizes, whereas a Demographic Clustering model can contain one cluster that contains 60% or 80% of the records together with small clusters containing 0.1% of the records. Therefore, Kohonen Clustering should not be used for Deviation Detection scenarios or other use cases in which the detection of small clusters is essential. Typical application scenarios for Kohonen Clustering are:

- Cross-marketing
- Cross-selling
- Customizing marketing plans for different customer types
- Deciding which media approach to use
- Understanding shopping goals
- Store profiling

Further reading

- *Neural Network Models, 2nd Edition*, by DeWilde

  The first 50 pages provide an overview on neural networks, and is a good compromise between understandability and detailed presentation.

- *Neural Networks in the Capital Markets*, by Apostolos-Paul, (Editor)

  The first 70 pages touch upon Kohonen maps and Backpropagation, data representation, and impact of parameters. Many useful case studies and references for financial applications.

- *An Introduction to Neural Networks*, by Anderson

  Covers many neural topics from biological information processing to technical applications. Has a good section on Gradient Descent Algorithms with Backpropagation and it also covers Kohonen maps.

- http://cvor.pe.wvu.edu/faq/faq-nn38.htm

  Contains further annotated references to neural literature.

- Kohonen, “Self-Organization and Associative Memory”, vol. 8 of Springer Series in Information Science, 3rd edition, Springer-Verlag, Berlin, 1989

  The classic paper on neural clustering.

Scoring a clustering model

Once a clustering model has been created and visually checked, you might want to apply it to either the data on which the model was created or to new data. Applying clustering models to a table means predicting for each record the best matching cluster ID and optionally other measures, such as the quality or confidence of the best match.

For this purpose, the Design Studio provides the Scorer operator. This operator takes two inputs: the model and the table to which the model is to be applied. In Figure A-6, we have connected a Scorer operator to a clustering model that groups the customers of a bank. The model performs the grouping based on demographic data (such as age or profession) and transaction statistics (such as the average balance on the customer's account). The table matches with the model and provides this required information.

In the Properties view of the Scorer operator, you can select which scoring results will be generated. By default, for each data record the ID of the best matching cluster and the quality of the match, a value between 0 and 1 is generated. In Figure A-6, we have also activated the confidence measure. This measure compares the qualities of the best matching cluster and the second-best matching cluster and answers the question: “How sure can we be that the best matching cluster is indeed the correct one?”.

The exact formula and meanings of the quality measure depends on the clustering algorithms that have been used to created the clustering model. We will explain the exact formulas for Demographic and Kohonen Clustering at the end of this section.

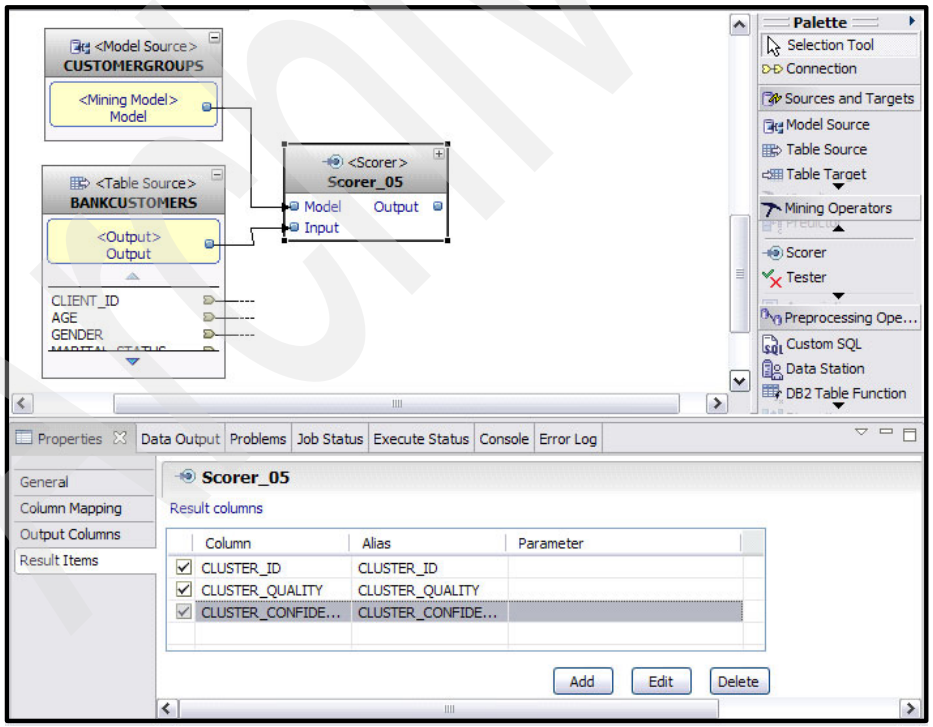


Figure A-6 Properties View: Scorer Operator

Now we have to provide a table into which the scoring results will be written. By default, the scorer operator tries to write all columns of the input table plus the activated scoring result measures into the result table. If you do not want to reproduce the input table content in the result, you can deactivate certain output columns in the Output Columns window of the Scorer's Properties view. In Figure A-7, we are removing all input columns except the primary key column CLIENT_ID from the result table.

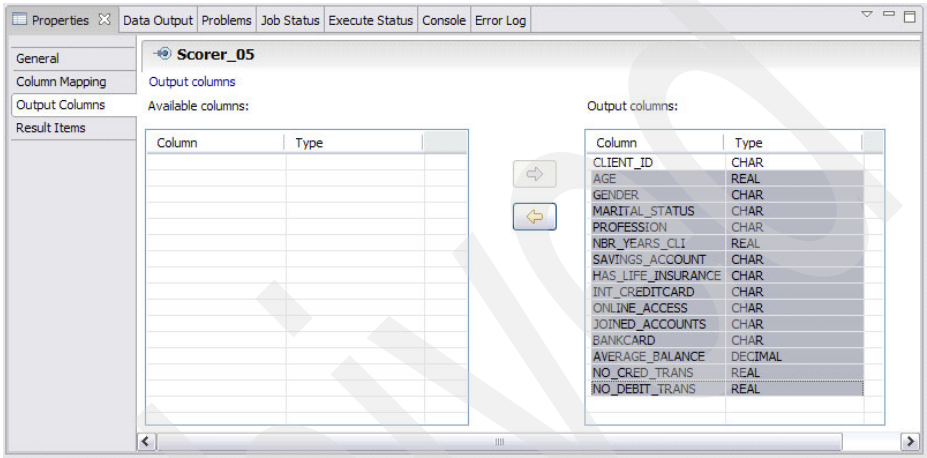


Figure A-7 Scoring results table

In many cases, a result table of exactly matching column format does not yet exist. Instead of manually creating a new table, you can right-click the output port of the Scorer operator and select the menu item **Create suitable table**. A wizard pops up in which you are presented with a set of default settings for the new table. You can modify these default settings. In Figure A-8, we have marked the column CLIENT_ID of the new table as primary key. All other settings are the default settings.

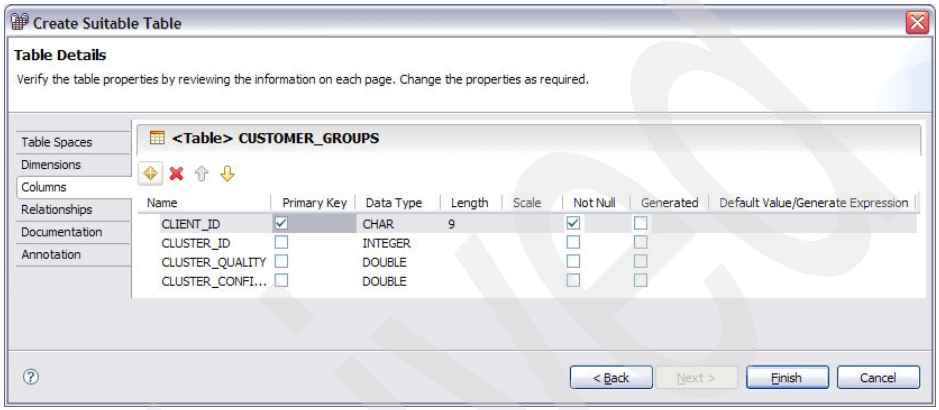


Figure A-8 Results table details

Now we can click the **Finish** button of the table creation wizard, and the resulting mining flow has the desired form, as depicted in Figure A-9. We can execute this flow and fill the table CUSTOMER_GROUPS with the desired scoring results.

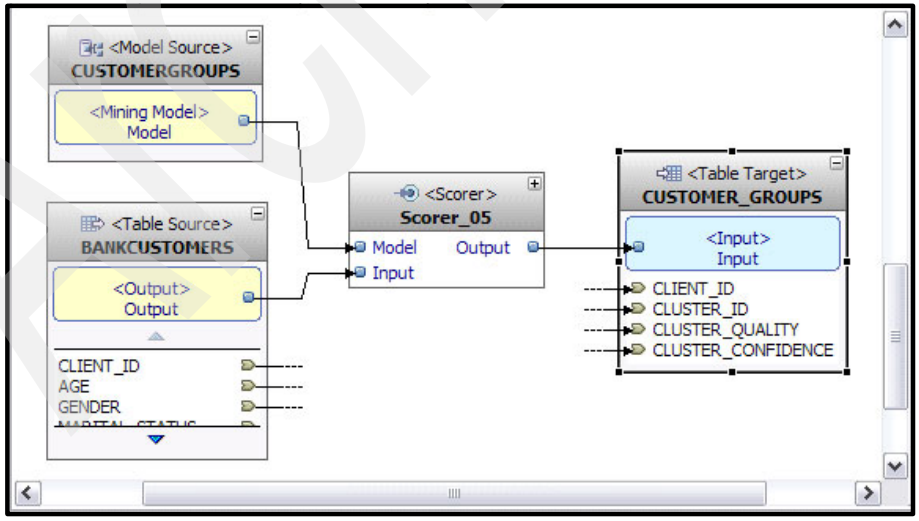


Figure A-9 Graphical mining flow result

Basic scoring result measures

The graphical windows of the Scorer operator allow modifying and activating the most commonly used result measures of clustering scoring.

Cluster ID

This is the ID of the best matching cluster. Clusters in a Clustering model are labeled by IDs running from 1 to the number of clusters in the model.

Cluster Quality

The quality Q(x,C) of a record x with a cluster C is a number between 0 and 1, which indicates how well the record fits into the cluster C. From the DesignStudio, only the quality of the best-matching cluster, Q1 := $Q(x,C_{best})$, can be activated. Using the DB2 Warehouse SQL API and a custom SQL operator, the qualities for all other clusters can also be calculated.

For Demographic Clustering models, Q(x,C) is calculated in two steps. First, a score score(x,C) is calculated based on the percentage of positive votes pro(x,C) for putting record x in cluster C. For the exact definition of pro(x,C), we refer to the description of the Demographic Clustering algorithm:

score(x,C) := pro(x,C) / (pro(x,C) + contra(x,C))

In a second step, the score is compared to the best possible score for this cluster, which can be interpreted as the score of the cluster's imaginary center record:

Q(x,C) := score(x,C) / score(center(C), C)

For Kohonen Clustering models, we also first calculate a score. But here, the score measures the distance between the best matching cluster's center vector and the current record in the space of normalized input fields:

score(x,C) := dist(x, center(C))

Let N be the number of normalized input fields and M the number of clusters. Q(x,C) is then calculated as:

$Q(x,C) := 1 - (dist(x,center(C)) / avgDist(N,M))^{\sqrt{N}}$

If this formula returns a negative value, we set Q(x,C) = 0. Here, avgDist(N,M) is the average distance between a randomly chosen N-dimensional data point and the nearest out of M randomly chosen N-dimensional cluster center points. DB2 Warehouse uses an approximation formula for avgDist(N,M) that has been constructed using Monte-Carlo simulations. The formula for Q(x,C) is constructed in a way that a record that exactly hits the center position of its best matching cluster gets the quality 1, and a record that is as far away from its nearest cluster center as a random point from the nearest cluster center of a

random model get the quality 0. The exponent of $\sqrt{N}$ that appears in the formula can be explained as follows: avgDist has been defined such that the probability of finding a random cluster center point in the N-dimensional sphere of radius r=avgDist around any given point p is 1.

That means the average probability of finding a cluster center point in the N-dimensional sphere of radius r<avgDist around a given data point is $1 (r/avgDist)^N$ (because the volume of an N-dimensional sphere with radius r is proportional to r^N). That means if we find a cluster center at r<avgDist, the probability that this is not just random but something meaningful is $1-(r/avgDist)^N$. For practical implementation, we have the problem that we only know a rough approximation for avgDist. For large number of dimensions N, the exponent (^{N}) would increase the approximation error in avgDist considerably, which could even lead to numeric overflow for N>100..200. Therefore, we use a damped exponent of $\sqrt{N}$ instead of N.

Cluster confidence

This measure compares the qualities of the best matching cluster and the second-best matching cluster, Q_1 and Q_2.

For Demographic Clustering, the formula is:

confidence := $Q_1 / (Q_1 + Q_2)$.

And for Kohonen Clustering, it is:

confidence := $0.5 + 0.5 (1 - (dist(d,nearest) / dist(d,2ndNearest))^N)$.

Here, we can use the un-damped exponent (^{N}) for the Kohonen clustering formula instead of the damped exponent ($^{\sqrt{N}}$) because we use the exactly known denominator dist(d,2ndNearest) and not the approximated value avgDist.

In both cases, the meaning of confidence is “If I have the choice of assigning the current record either to the best matching or to the second best matching cluster, how sure can I be that selecting the best matching cluster is the correct choice?”

Advanced scoring result measures

Using 'Custom SQL' and the DB2 Warehouse SQL Mining API, you can access more advanced scoring results in addition to the ones described above. The Mining API defines a DB2 user defined type (UDT) called DM_ClusResultSpec that provides a set of methods for activating advanced scoring result measures:

- DM_getClusScore

  Returns the score score(x,C_{best}) as defined in “Basic scoring result measures” on page 364.

- DM_getClusScore(clusterID)

  Returns the score score(x,C) of the cluster with ID clusterID.

- DM_getQuality(clusterID)

  Returns the matching quality Q(x,C)) of the cluster with ID clusterID.

- DM_getClusScore

  Returns the score score(x,C_{best}) as defined in “Basic scoring result measures” on page 364.

The basic result measures described in “Basic scoring result measures” on page 364 are accessible in the API through:

- DM_getClusterID
- DM_getQuality
- DM_getClusConf

In Example A-1, we show an SQL code that could be embedded into a Design Studio workflow by means of the Custom SQL operator. The code is calculated for each data record, the quality of which matches with cluster 3 of the clustering model named 'CustomerGroups'.

Example: A-1 Custom SQL operator

```
WITH "RESULTSPECIFICATION"( "RESULTSPEC" ) AS
  ( VALUES( IDMMX.DM_ClusResultspec()..DM_setCluster(3) )),
  "CLUSTERVIEW"( "CLIENT_ID", "CLUSTERRESULT" ) AS (
    SELECT "CLIENT_ID",
    IDMMX.DM_applyClusModel (
      'CustomerGroups',
      rec2xml( 1.0, 'COLATTVAL', '',
       B."AGE",B."GENDER",B."MARITAL_STATUS",B."PROFESSION",
       B."BANKCARD",B."NO_CRED_TRANS",B."NO_DEBIT_TRANS" ),
      R."RESULTSPEC"
    )
    FROM "BANKING_SCORING" B, "RESULTSPECIFICATION" R
  )
 SELECT
  "CLIENT_ID",
  CAST( IDMMX.DM_getQuality("CLUSTERRESULT",3) AS DEC(5,4))
    AS "QUALITY_FOR_CLUSTER_3"
 FROM "CLUSTERVIEW";
```

There are three commands of the Mining API. The first one, DM_setCluster(3), specifies that we want to obtain the matching quality with cluster 3 for each data record. The second command, DM_applyClusModel, reads one data record and calculates its scoring results. The content of this record is provided through the blue rec2xml statement. The third API command, DM_getQuality(…,3), retrieves the matching quality of cluster 3 for the current record.

Classification

Classification is the process of automatically creating a model of classes from a set of records that contain class labels. The classification technique analyzes records that are already known to belong to a certain class, and creates a profile for a member of that class from the common characteristics of the records. You can then use a data mining scoring tool to apply this Classification model to new records, that is, records that have not yet been classified. This enables you to predict if the new records belong to that particular class. Commercial applications of this mining function include credit card scoring, ranking of customers for directed mailing, the prediction of credit risk in new customers, and attrition prediction.

The dependant field (predicted field) y of a Classification model is always categorical, often binary, for example yes/no, high/low, or "0"/"1". The independent fields $x = (x_1,\ldots,x_n)$ can be categorical or numeric.

A vast set of classification algorithms has been proposed and successfully applied to certain use cases. One important group of classification algorithms are the so-called linear classifiers. A linear classifier defines a hyperplane $w \bullet x = \sum_{i=1..n} w_i x_i$ in the n-dimensional space of dependent data fields and calculates the predicted class label y as $y = f(w \bullet x)$. Popular linear classification algorithms are:

- Linear Discriminant Analysis (LDA), which assumes a Gaussian conditional density function P(x|y).
- Naïve Bayes Classification, which assumes an independent binomial conditional density function P(x|y).
- Logistic Regression, a maximum likelihood estimation of w assuming that the observed training set was generated by a binomial model that depends on the output of the classifier.
- Support Vector Machine, an algorithm that maximizes the minimum distance between the decision hyperplane and the data records in the training set.
- Perceptron, an artificial neural network.

Linear classification algorithms are robust and fast, but they systematically neglect product-like factors such as $x_i \bullet x_i$ or $x_i \bullet x_j$. Examples for non-linear classifiers are:

- Decision Tree
- Several Neural Classification algorithms

DB2 Warehouse offers the linear classifiers Naïve Bayes and Logistic Regression, and the non-linear classifier Decision Tree.

Tree Classification

A decision tree is a class discriminator that recursively partitions the training set by binary splits until each partition consists entirely or dominantly of examples from one class.

Algorithm

Each interior node corresponds to an independent variable; an arc to a child represents a possible value of that variable. Each leaf represents a possible value of the dependent variable given the values of the independent variables represented by the path from the root. Each non-leaf node of the tree contains a split point that is a test on one or more dependent field values and determines how the data is partitioned.

Figure A-10 on page 369 shows an example for a decision tree. The data consist of the two independent fields Age and Services. These fields describe customers of a telecommunications company. For each customer, the company wants to predict whether or not he or she is likely to leave the company in favor of a competitor. Hence, the dependent field is the field Churn with the two values STAY and LEAVE. The left graph of the picture shows data from the past with known churn behavior. The customers who stayed are represented by green points; those who left by red points. The right part of the picture shows a decision tree that has been created based on these historic data. The tree contains the rules that those customers who are younger than 30 and have subscribed for less than three services are likely to leave; all others are likely to stay.

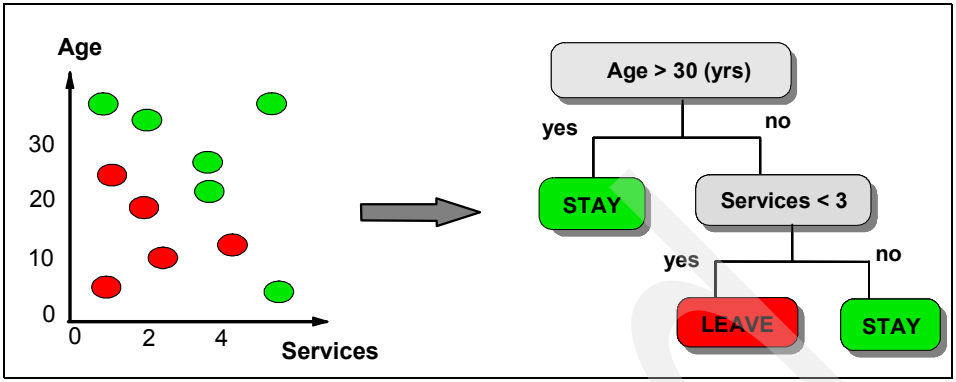


Figure A-10 Sample decision tree

The tree is built by recursively partitioning the data. Partitioning continues until each partition is either pure (all members belong to the same class) or sufficiently small (a parameter set by the user). The entire initial set of training data records is associated with the root of the decision tree. As the tree is grown and nodes are split to create new children, the data records associated with the parent node are partitioned and each record is associated with one of the children.

A decision tree classifier is built in two phases. In the initial growth phase, a preliminary tree is built. In the second phase, the prune phase, a less complex sub-tree is created by cutting away and merging branches of the initial tree. Small, deep nodes of the tree resulting from noise contained in the training data are removed, thus reducing the risk of overfitting, and resulting in a more accurate classification of unknown data. While the decision tree is being built, the goal at each node is to determine the split attribute and the split point that best divides the training records belonging to that leaf.

The value of a split point depends on how well it separates the classes. Several splitting indices have been proposed in the past to evaluate the quality of the split. DB2 Warehouse Tree Classification uses the gini index. The gini index of a partition (P_1,P_2) of a set $S=P_1 \cup P_2$, with $|S|=n$, $|P_1|=n_1$, $|P_2|=n_2$, is:

$$G(P_1,P_2) = n_1/n \bullet (1-\sum_{i=1..m} p_1(i)^2) + n_2/n \bullet (1-\sum_{i=1..m} p_2(i)^2)$$

where m is the number of different class labels and $p_j(i)$ is the relative frequency of class label i in partition P_j. The gini index reaches its minimum of zero when all cases in both parts of the partition fall into a single target category (class label). Therefore, the gini-based splitting criterion is to minimize $G(P_1,P_2)$.

After building the decision tree, the tree is pruned using the so-called Minimum Description Length (MDL) strategy. MDL strategy takes into account the:

- Misclassification cost produced by the model when classifying the training data
- Cost of describing the model in terms of split condition complexity

MDL demands minimizing the sum of both costs:

- Nodes with complex split condition causing small number of errors when pruned are removed.
- Pruning is supposed to reduce the risk of overtraining and to improve model reliability.

The Tree Classification contains several refinements of the basic algorithm described above.

First, in many use cases it is practically impossible to try all possible partitions (P_1,P_2) for defining a new split. Therefore, a heuristic is used to rule out many possible partitions before actually computing their gini index.

Second, in use cases with very asymmetrically distributed target values, a biased optimization criterion can be more suitable. For example, when trying to predict fraudulent credit card transaction out of a training data set with 1% frauds and 99% normal transactions, a trivial decision tree that always predicts normal would get excellent gini index ratings, but the resulting model would be completely useless. Tree Classification contains some default adjustment for asymmetrically distributed target values. In addition to this, the user can further modify the optimization criterion by defining a so-called misclassification cost matrix. In such a matrix, the user could specify, for example, that misclassifying a fraudulent transaction as normal should be regarded 100 times as bad or costly as misclassifying a normal transaction as fraud. This compensates for the effect of asymmetrically distributed target values.

Third, the user can define hard limits for the maximum tree depth, the minimum leaf node size, and the maximum purity of the internal tree nodes.

Parameters and advanced options

In the DesignStudio, Tree Classification functionality is called through the Predictor operator. The Predictor operator in its default state is algorithm agnostic. Once you have defined the target field in the MiningSettings window of the operator's Properties view, the operator becomes a Tree Classifier (if the target field is categorical) or a Transform Regression Numeric Predictor (if the target field is numeric). Once a categorical target field has been selected, the following advanced options are directly supported by the Mining Settings window of the Predictor operator in the DesignStudio.

Maximum purity

This option specifies the maximum allowed purity of internal tree nodes. Allowed values are percentage values between 0 and 100; 100 is the default. If a value x<100 is specified, all non-leaf nodes for which more than x% of the associated training data records have one single target value (or class label), are pruned and become leaf nodes.

Typing a value x into the field Maximum purity is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setAlgorithm('Tree','<MaxPur>x</MaxPur>').

Maximum depth

This option is used to define an upper limit for the tree depth. The depth of a tree node is the number of edges between this node and the root node of the tree. If an integer value x is specified for maximum depth, all leaf nodes with depths larger than x are pruned. Per default, the maximum tree depth is unlimited.

Typing a value x into the field Maximum depth is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setAlgorithm('Tree','<MaxDth>x</MaxDth>').

Minimum number of records

This option is used to define a lower limit for the number of records in a single tree node. If an integer value x is specified for Minimum number of records, all nodes containing less than x records are pruned and merged into their parent nodes. By default, the minimum number of records per tree node is five.

Typing a value x into the field Minimum number of records is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setAlgorithm('Tree','<MinRec>x</MinRec>').

Field Usage Types

In the Column Properties page of the Predictor Properties view, each available input data field has a default usage type System determined. That means the classification kernel is free to decide which fields it will use as active fields, and which ones are not useful because they are either (almost) single-valued or (almost) key-like or highly correlated with another field. Only active fields can be used as splitting conditions in tree nodes.

You can overwrite the usage type System determined for any data field if you click the value **System determined** in table column Field Usage Type. A menu pops up from which you can select either **active** or **inactive**. Active fields will be used for defining the tree's split conditions. Inactive fields are not used and not even read during the mining; the mining kernel completely neglects the corresponding input data columns.

Modifying the column Field Usage Type in the Column Properties window is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setFldUsageType('fldName',x). If the first argument 'fldName' is missing, the command alters the default usage type for all fields: DM_setFldUsageType(x). Allowed values for x are 1 (active) or 2 (inactive).

Field Types

The Column Properties view of the Predictor Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields, whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one categorical, which means textual, value, and no ordering such as 'x < y' is assumed between different values. In the resulting tree model, the field's statistics is stored and visualized in the form of a categorical pie chart, and not as a numeric histogram.

Alternatively, the field type can be set to categorical by typing the following command into the Optional Parameters free text field: DM_setFldType('fldName',0).

The following advanced options do not have a directly corresponding input field in the Predictor operator properties view. You have to type the API command into the general purpose Optional Parameters field on the Mining Settings window.

Figure A-11 shows this for the parameter Cost Matrix, together with the power option -IDM_MAX_DISCR_COUNT.

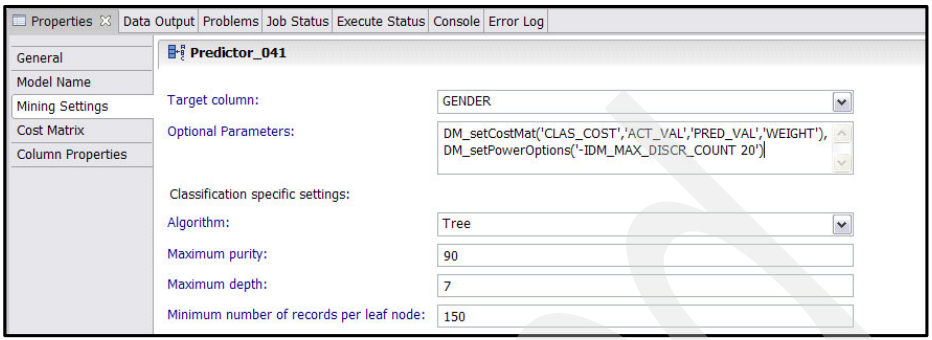


Figure A-11 Predictor Operator Properties View

Cost matrix

Sometimes, you might obtain a model that misclassifies the less frequent target values while achieving a very low overall error rate. To get better models with very skewed data, the tree heuristic automatically generates an appropriate cost matrix to balance the distribution of class labels when a decision tree is trained. You can also manually adjust the cost matrix.

A cost matrix (error matrix) is also useful when specific classification errors are known to be more severe or more costly than others. The Classification mining function tries to avoid classification errors for which a high error weight has been specified in the cost matrix. The trade-off of avoiding expensive classification errors is an increased number of cheap classification errors. Thus, the overall number of errors increases while the cost of the errors decreases in comparison with the same classification without a cost matrix. Weights specified must be greater than or equal to zero. The default weight is 1. The cost matrix diagonal must be zero. You can assign error weights to misclassifications by referencing a preexisting DB2 table which contains misclassification weight factors.

For example, in the credit card fraud scenario described above, Table A-3 might have been created.

Table A-3 Fraud Table Example

ACT_VAL	PRED_VAL	WEIGHT
fraud	normal	50

This states that misclassifying a fraudulent transaction as a normal one should be regarded 50 times as severe as misclassifying a normal transaction as fraudulent. Now we can activate it for Tree Classification by typing the following command into the Optional Parameters text field of the Predictor operator:

```
DM_setCostMat('CLASS_COSTS','ACT_VAL','PRED_VAL','WEIGHT')
```

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within 'clause' must be surrounded by two single quotes. Here is an example:

```
DM_setWhereClause('GENDER=''m''').
```

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field 'fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

Tree Classification supports all power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the Tree Classification kernel.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.
- -IDM_MAX_DISCR_COUNT and its variants specify how many different valid values a discrete field, for example a categorical field, can have. This power option is particularly important for Tree Classification. The default value is 100, which means for each categorical field the first 100 different values found in the training data are regarded as valid values, while all other values are regarded as invalid. Increasing the threshold to a value larger than 100 can increase the quality of the generated model, but it might considerably slow down the training process. On the other side, reducing the threshold, for example to 50 or 20, can considerably speed up the training process.

Content of a Tree Classification model

Tree Classification models are stored as PMML <TreeModel> in the repository table IDMMX.CLASSIFMODELS. They contain the following information.

Model quality

Each Tree Classification model contains a model quality number Q in the range [0…1]. Q is calculated from two or three detailed quality measures A, R, and Rel:

- Accuracy A = nbCorrectClassifications / nbTrainingOrTestRecords.
- Ranking Quality R = area between a model's gains chart and a random model's gains chart / area between the best possible gains chart and a random model's gains chart.
- Reliability Rel := A(calculated on training data) / A(calculated on test data). The reliability measure can only be calculated if test data with known class label are available that have not been used to train the model.
- Model quality Q is the arithmetical mean of the available detailed quality numbers:

  Q = (A+R+Rel)/3 or Q = (A+R)/2.

If test data is available, for example if the model is obtained from the output port of a Tester operator in the DesignStudio, A and R are calculated on the test data. If no test data is available, for example if the model is obtained from the output port of a Predictor operator in the DesignStudio, A and R are calculated on the training data, and in this case their values should be regarded with caution because the model might be overfitted and its quality indicators on the training data might not correctly represent the model's predictive power on unknown data.

Field information for active fields

For each input data field that has been used as an active field in the classification, the model contains the following measures:

- Value distribution: Frequencies of the discrete values for categorical fields; frequencies of discretized histogram bins for numeric fields. The bin boundaries are automatically chosen by the classification kernel in a suitable way, such that between about 15 and 80 bins with reasonably spaced and rounded boundary values are created.
- Field importance indicator: A value in the range [0…1] indicating the fraction of the model's total predictive power that is contributed by the current field. The sum of all fields' importances equals 1. Importance factors are calculated in several steps.

First, each internal tree node P (with child nodes C_1 and C_2) is assigned a classification power:

$pow(P) = |P| \bullet (1 - |P| \, gini(P) / (|C_1| \, gini(C1) + |C_2| \, gini(Ch_2)))$

where pow(P) describes the extent by which node P's split increases the purity of the resulting partitions.

Second, for each field, all internal tree nodes {P} involving this field in the split criterion are identified and their classification powers are summed up, resulting in a raw importance number of the field.

Third, all raw importance numbers are rescaled so that the sum of all field importances equals 1.

- Modal value and modal frequency.

Field-field correlations

The Tree Classification model contains field-field correlation coefficients for a subset of all possible field-field pairs (f,g) between active fields f and g.

- If f and g are continuous numeric, DB2 Warehouse calculates linear correlation coefficients:

  $corr(f,g) := \sum_{i=1...N}(f_i-\mu_f)(g_i-\mu_g) / N\sigma_f\sigma_g$

  where (μ_f,σ_f) and (μ_g,σ_g) are mean and standard deviations of the values of fields f and g, and N is the number of data records.

- If f and g are discrete, that means discrete numeric or categorical, DB2 Warehouse calculates adjusted contingency coefficients:

  $cont(f,g) := \sqrt{(k/(k-1) \bullet \chi^2 / (N+ \chi^2))}$

  where χ^2 is the result of the χ^2 test that tests the hypothesis that the value of f significantly influences the value distribution of g (or vice versa). If f has k_f different values and g has k_g different values, then k is defined as $k := \min(k_f,k_g)$.

- No correlation is calculated between continuous and discrete fields.
- Furthermore, no correlations are calculated for fields that the mining kernel has removed from the list of active fields because they are either (almost) single valued or (almost) key-like.

Tree node statistics

The Tree node can have the following values:

- Value distributions of all active fields for all records associated with the current tree node (both for internal and for leaf nodes)
- Modal value and modal frequency for each active field and each node

Class label statistics

The Class label can have the following values:

- Value distributions of all active fields for all records having the current class label (target field value) tree node
- Modal value and modal frequency for each active field and class label

Gains charts

A gains chart (or lift chart) is an algorithm-independent visualization and comparison method that allows you to measure and compare the abilities of classification models to reliably separate the data records with high probability of having a certain class label from those ones for which the independent field values indicate a low probability of having that class label.

A Tree Classification model contains a separate gains chart for each target field value (class label). The gains chart for class label L is a two-dimensional plot in which on the x-axis, all N_{rec} available training or test data records are summed up, ordered by decreasing predicted confidence that the record has the class label L. Hence, the x-axis runs from 0 to N_{rec}. On the y-axis, the gains chart tracks the accumulated number of records between x=0 and the current x-axis position that actually have the class label L. Therefore, each gains chart plot starts at point (0,0) and ends at point (N_{rec},N_{rec}(L)), where N_{rec}(L) is the total number of data records having class label L. The more a model's gains chart plot deviates in an upward direction from the straight line between these two points (the so called 'random model line'), the better is the model.

Figure A-12 shows a gains chart for the value M of target field GENDER. The blue line is the model's actual gains chart, the dashed red line is the random model line, and the dotted green line is the best achievable gains chart on these input data.

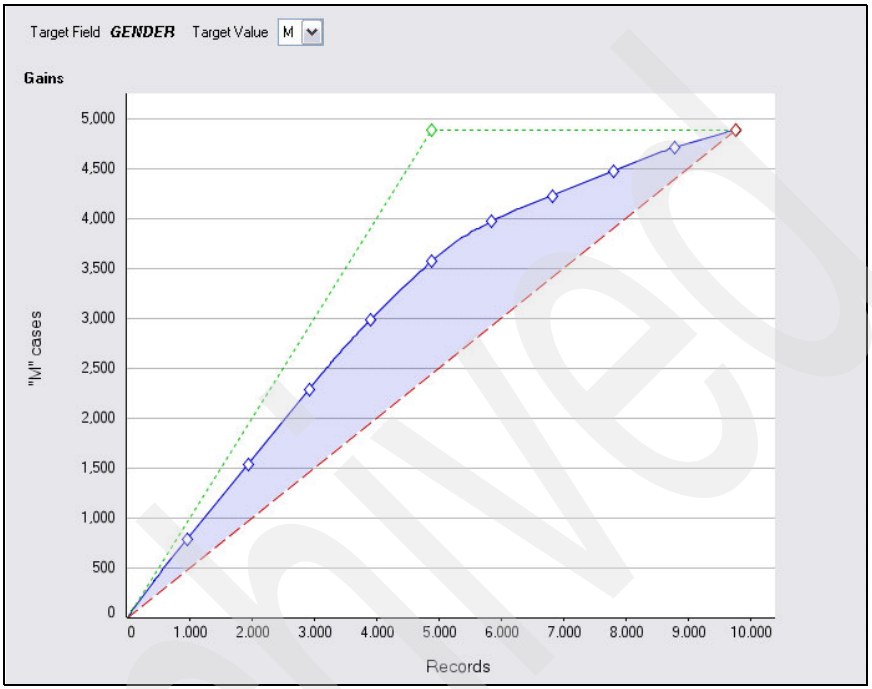


Figure A-12 Gains Chart

Confusion matrix

The confusion matrix, depicted in Figure A-13, is an overview of the predictive accuracy of the model. It compares actual and predicted class labels and shows which misclassifications the model has made on the training or test data.

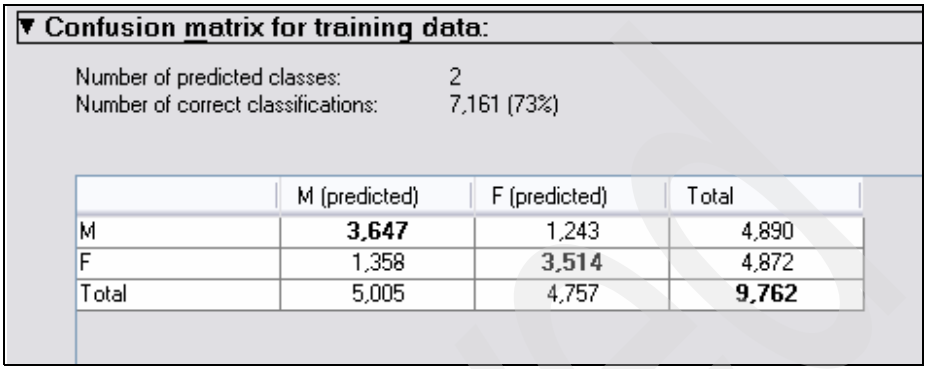


Figure A-13 Confusion Matrix

Methods and operators

This section discusses methods and operators for extracting model content into DB2 tables.

Tree Rules Extractor operator

The mining flow editor of the Design Studio Design Studio contains an operator, Tree Rules Extractor operator, that creates a table containing the following columns:

- NODEID (varchar(128)): Node ID of the current node. The node ID is a series of digits 1 or 2, separated by dots (.). 1 is the root node, 1.1 is the first, 1.2 the second child node of the root node. 1.2.2.1 is the first child of the second child of the root node's second child node.
- ISLEAF (smallint): A Boolean flag (0/1) that indicates whether the current node is a leaf node or an internal node.
- CLASS (varchar(256)): The predicted value of the current node.
- CONFIDENCE (real): Confidence of the current node's prediction, calculated from the node's purity on the training data set. The value range is [0…1].
- SIZE (bigint): Number of data records associated with the current node.
- DEPTH (smallint): Depth of the current node inside the tree. The root node has depth 1, the node 1.2.2.1 has depth 4.
- CONDITION (varchar(2048)): A textual description of all consecutive split conditions that lead to the current node.

Gains Chart Extractor operator

The mining flow editor of the Design Studio contains an operator 'Gains Chart Extractor' that writes the gains chart data for one selected target value L into a table containing the following columns:

- THRESHOLD (double): Minimum confidence threshold of the current gains chart point. The value range is [0…1].
- ROWCOUNT (bigint): The accumulated number of all test or training data records that have a predicted confidence of at least THRESHOLD to have target value L. This is the x-coordinate of the current point in the gains chart.
- SUMACTUAL (double): The accumulated number of all test or training records that actually have the target value L and for which the predicted confidence to find target value L is at least THRESHOLD. This is the y-coordinate of the current point in the gains chart.

Quality Extractor operator

The Quality Extractor, which can also be used with any other DB2 Warehouse mining model, writes four quality numbers into a four-column table:

- MODELQUALITY (real): Overall model quality Q as defined above
- PREDICTIONACCURACY (real): Accuracy A as defined above.
- RANKINGQUALITY (real): Ranking quality R as defined above.
- RELIABILITY (real): Reliability quality Rel as defined above.

SQL API commands and Custom SQL operator

The DB2 Warehouse mining SQL API offers some more model extractor methods for which no specific graphical mining flow operator exists. These commands can be used within a mining flow by means of the Custom SQL operator. Figure A-14 shows how the API command DM_getConfMatrix is called within a Custom SQL operator. The screen capture shows the Properties view of a Custom SQL operator, which has been drawn to the DesignStudio' Mining Flow Editor canvas and whose input port has been connected to the table source 'CONF_MATRIX'.

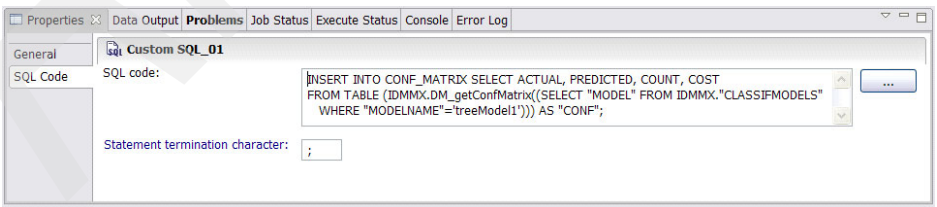


Figure A-14 Custom SQL operator

This operator writes the model's confusion matrix and cost matrix into the table CONF_MATRIX consisting of the columns ACTUAL_VALUE, PREDICTED_VALUE, COUNT, and COST.

The following model introspection commands for Tree Classification models, which means objects of type IDMMX.DM_ClasModel, are available:

- IDMMX.DM_expClasModel(clasModel)

  Returns the PMML model as a flat XML file in the form of a CLOB.

- IDMMX.DM_getClasTarget(clasModel)

  Returns the name of the target field.

- IDMMX.DM_getMdlQuality(clasModel)

  Returns the model quality Q.

- IDMMX.DM_getReliability(clasModel)

  Returns the reliability quality Rel.

- IDMMX.DM_getRankQuality(clasModel,classLabel)

  Returns the ranking quality R for the class label classLabel.

- IDMMX.DM_getPredAccuracy(clasModel,classLabel)

  Returns the fraction of records that actually have the class label classLabel and for which this class label is also the predicted one.

- IDMMX.DM_getTreeRules(clasModel)

  This command is the API equivalent of the Cluster Extractor operator. It returns a 7-column table (NODEID, ISLEAF, CLASS, CONFIDENCE, SIZE, DEPTH, and CONDITION) in which each row describes one internal node or leaf node contained in the tree model.

- IDMMX.DM_getGainsChart(clasModel,classLabel)

  This command is the API equivalent of the Gains Extractor operator. It returns a 3-column table (ROWCOUNT, SUMACTUAL, and THRESHOLD) in which each row describes one point of the gains chart for the class label classLabel. ROWCOUNT is the x-coordinate of the point, SUMACTUAL the y-coordinate.

- IDMMX.DM_getConfMatrix(clasModel)

  This command returns a 4-column table (ACTUAL, PREDICTED, COUNT, and COST) in which each row describes one non-diagonal matrix element of the tree model's confusion and cost matrix. Non-diagonals not contained in the output of this function have a COUNT of 0 and a COST of 1.

- IDMMX.DM_getQualities(clasModel,classLabel)

  This command is the API equivalent of the Qualities Extractor operator. It returns a 2-column table (QUALITYNAME and QUALITYVALUE) in which each row contains one of the quality numbers model quality Q, accuracy A, ranking quality R, or reliability Rel.

- IDMMX.DM_getFields(clasModel)

  Returns a 4-column table (COLNAME, FIELDNAME, MININGTYPE, and IMPORTANCE) that contains for each active field in the model the name of the data column to which it refers, its alias name, its mining type (0 for categorical or 1 for numeric), and its importance, a value between 0 and 1. The sum of all importance values is 1.

- IDMMX.DM_getCorrelations(clasModel)

  Returns a 3-column table (FIELDNAME1, FIELDNAME2, and CORRELATION) that lists all field-field correlation numbers contained in the model. Correlations between continuous numeric fields are linear correlation coefficients, correlations between discrete fields are adjusted contingency coefficients, and correlations between discrete and continuous fields are not calculated.

Applications of Tree Classification

Tree Classification is a robust standard classification method that can be applied to data and use cases in almost any industry, for example:

- Approving or denying insurance claims
- Detecting credit-card fraud
- Identifying defects in manufactured parts
- Diagnosis of error conditions
- Defining optimum customer groups for target marketing
- Medical diagnosis, risk assessment, and medical treatment effectiveness testing
- Inventory replenishment
- Store location planning

Compared to most other classification techniques, Tree Classification has the advantage to produce not only predictions but also easily understandable explanations and rules why a specific prediction was made. For example, in the churn risk example shown in the algorithm description above, for a 28 year old customer with two service subscriptions, we would not only get the prediction “high churn risk”, but also the explanation “if a customer is less than 30 years old and has subscribed for less than three services, then he or she is likely to churn”. This explanatory power of Tree Classification is probably the main reason for the popularity of this approach.

The method can work both with predominantly categorical and with predominantly numeric data. However, for data with many categorical fields or with categorical fields with many different values, runtimes, and memory consumption might be considerably higher than with other classification algorithms, such as Naïve Bayes Classification.

Further reading

- http://en.wikipedia.org/wiki/Category:Classification_algorithms
- "Classification and regression trees" by L. Breiman, et al, in Wadsworth, 1984
- "Parallel Classification for Data Mining on Shared-Memory Multiprocessors" by M. J. Zaki, et al, 1996
- "SPRINT: A Scalable Parallel Classifier for Data Mining", by J. Shafer, et al, 1996.
- Introduction to contingency tables:

  http://en.wikipedia.org/wiki/Contingency_table

Naïve Bayes Classification

In this section, we discuss the Naïve Bayes Classification algorithm.

Algorithm

The starting point for the Naïve Bayes Classification algorithm is a probability model $p(t_k|f_1,\ldots,f_n)$, where f_1, …, f_n are the independent fields, t_k are the target values (class labels), and p(y|x) stands for conditional probability that y is true if we know that t_x is true. Based on this probability model, a prediction of t_k for given independent field values f_1, …, f_n can be calculated as:

$$k_{predicted} = \max_k(p(t_k|f_1,\ldots,f_n)).$$

That means we always predict the target value that has the highest conditional probability for the given independent field values. The task now is to efficiently define and store a suitable probability model. To that purpose, we first rewrite the expression $p(t_k|f_1,\ldots,f_n)$ using Bayes' theorem of conditional probability:

$$p(t_k|f_1,\ldots,f_n) = p(t_k) \quad p(f_1,\ldots,f_n|t_k) / p(f_1,\ldots,f_n).$$

The denominator $p(f_1,\ldots,f_n)$ is not a function of t_k and can be ignored when searching for the maximum of $p(t_k|f_1,\ldots,f_n)$:

$$k_{predicted} := \max_k(p(t_k) \quad p(f_1,\ldots,f_n|t_k)).$$

The joint probability distribution $p(f_1,\ldots,f_n|t_k)$ is still nothing we can compute and store in a numerically efficient way. Therefore, we make the “naïve” assumption that the independent field values f_i and f_j are all conditionally independent for $i \neq j$. In this case, the joint conditional probability distribution factors into a product of atomic conditional probabilities, and we obtain the prediction formula:

$$k_{predicted} = \max_k(p(t_k) \quad p(f_1|t_k) \quad \ldots \quad p(f_n|t_k))$$

The simple conditional probabilities $p(f_i|t_k)$ can be calculated very easily if all co-occurrence counts $count(f_{i(j)} \wedge t_k)$ on the training data are collected:

$$p(f_{i(j)} \mid t_k) = count(f_{i(j)} \wedge t_k) / count(t_k)$$

Here, $count(f_{i(j)} \wedge t_k)$ is the number of training data records in which the independent field i assumes its field value with index j, and simultaneously the target field assumes the target value with index k. Therefore, a Naïve Bayes model is simply a 3-dimensional data structure of co-occurrence counts $c(i,j,k) := counter_{i(j)} \wedge t_k)$, plus a univariate value statistics of target field values, $count(t_k)$.

Implicitly, these formulas assume that all independent fields f_i are discrete. If some of the actual fields of the input data are continuous numeric, these fields must be discretized into a finite number of intervals in a preprocessing step.

- How realistic is the “naïve” assumption of conditional independence of all independent field values? In most real-world data, it is completely unrealistic. For example, in demographic data, average income will be higher with increasing age or with higher education; certain home location types, hobbies, or interests are correlated with gender, social status, profession or education level, and so on. Does this mean that Naïve Bayes models are often of low quality and have little predictive power on real-world data? Surprisingly not. One intuitive reason for this robustness of the method with respect to violations of the underlying independence assumption is the fact that the method still gives the correct prediction if the involved joint conditional probabilities are significantly misjudged but their relative ordering, which is the only information that enters into the calculation of max_k, remains correct. More detailed studies and justifications of the method's robustness can be found in the literature, for example in "An empirical study of the naive Bayes classifier", found at:

  http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf

Parameters and advanced options

In the DesignStudio, Naïve Bayes Classification is called through the Predictor operator. The Predictor operator in its default state is algorithm agnostic. Once you have defined the target field in the MiningSettings window of the operator's Properties view, the operator either becomes a classifier (if the target field is categorical) or a numeric predictor (if the target field is numeric). After selecting a categorical target field, the algorithm Naïve Bayes can be selected in the drop-down menu Algorithm. The algorithm supports the following advanced options.

Probability threshold

This option specifies a minimum probability $p_{min}(f_{i(j)} \mid t_k)$ that is used instead of all probabilities $p(f_{i(j)} \mid t_k)$ that are smaller than the threshold, and in particular for those probabilities that are zero because the corresponding combination of $f_{i(j)}$ and t_k does not occur in the training data. The default value is 0.001.

Typing a value x into the field Probability threshold is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setAlgorithm('NaiveBayes','<ZeroProba>0.0001</ZeroProba>').

Field Usage Types

In the Column Properties page of the Predictor Properties view, each available input data field has a default usage type of System determined. That means the classification kernel is free to decide which fields it will use as active fields, and which ones are not useful because they are either (almost) single-valued or (almost) key-like or highly correlated with another field. Inactive fields are not used and not even read during the mining; the mining kernel completely neglects the corresponding input data columns.

Modifying the column Field Usage Type in the Column Properties window is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setFldUsageType('fldName',x). If the first argument 'fldName' is missing, the command alters the default usage type for all fields: DM_setFldUsageType(x). Allowed values for x are 1 (active) or 2 (inactive).

Field Types

The Column Properties view of the Predictor Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one categorical, which means textual, value, and no ordering such as 'x < y' is assumed between different values. Defining a numeric field as categorical in a Naïve Bayes Classification operator should be done with caution because it can significantly increase runtime and memory requirement of the algorithm: by default, Naïve Bayes Classification discretizes numeric fields in a small number, typically 20, of discrete intervals or bins. This discretization is suppressed when the field is declared categorical.

Alternatively, the field type can be set to categorical by typing the following command into the Optional Parameters free text field: DM_setFldType('fldName',0).

Multi-valued fields

For DB2 Warehouse versions higher then Version 9.1.2, a third field type called multi_categorical can be chosen on the Field Types field of the Column Properties window. If this field type is selected for a field, Naïve Bayes Classification expects to find XML strings of the form:

<item>$text_1$</item><item>$text_2$</item>…<item>$text_n$</item>.

Here, $text_1$ to $text_n$ are arbitrary categorical values. Each of them will be interpreted as one separate field value. The stop words <item> and </item> are not interpreted as parts of the field's values; they only serve to separate one value from the following one. Optionally, you can wrap the string <item>$text_1$ … $text_n$</item> into a framing <itemset>…</itemset> root tag, and you can also prepend an XML header string such as '<?xml version="1.0" encoding="xxx" ?>. This format has the advantage of being a well-formed XML document.

The format multi_categorical is particularly useful in connection with text analysis. For example, the analysis of a patient's medical history might have detected the keywords apnoe, smoker, or obesity. Then this text analysis result can be cast into one single multi-valued field as:

<item>apnoe</item><item>smoker</item><item>obesity</item>.

The following advanced options do not have a directly corresponding input field in the Predictor operator properties view. You have to type the API command into the general purpose Optional Parameters field on the Mining Settings panel. Figure A-15 shows this for the parameter WhereClause.

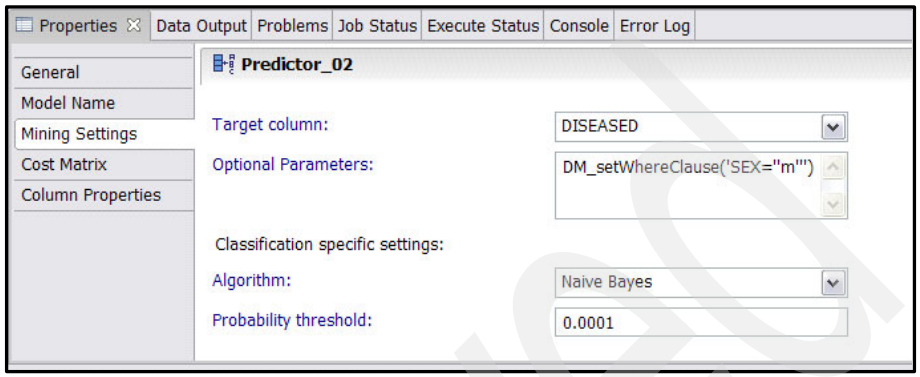


Figure A-15 Predictor operator properties view

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within 'clause' must be surrounded by two single quotes. Here is an example:

```
DM_setWhereClause('SEX=''m''').
```

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field 'fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

Naïve Bayes Classification supports all power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the Naïve Bayes Classification kernel.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.
- IDM_MAX_DISCR_COUNT and its variants specify how many different valid values a discrete field, for example a categorical field, can have. This power option can be important for Naïve Bayes Classification. The default value is 100, which means for each categorical field, the first 100 different values found in the training data are regarded as valid values, all other values are regarded as invalid. Increasing the threshold to a value larger than 100 can

increase the quality of the generated model, but it might considerably increase memory requirements during the training process and the size of the created model.

Content of a Naïve Bayes Classification model

Naïve Bayes Classification models are stored as PMML <BayesModel> in the repository table IDMMX.CLASSIFMODELS. They contain the following information.

Model qualities

Each Naïve Bayes Classification model contains a model quality number Q in the range [0…1]. Q is calculated from two or three detailed quality measures A, R, and Rel:

- Accuracy A = nbCorrectClassifications / nbTrainingOrTestRecords.
- Ranking Quality R = The area between a model's gains chart and a random model's gains chart divided by the area between the best possible gains chart and a random model's gains chart.
- Reliability Rel := A(calculated on training data) / A(calculated on test data). The reliability measure can only be calculated if test data with known class labels are available that have not been used to train the model.
- Q is the arithmetical mean of the available detailed quality numbers:

  Model quality Q = (A+R+Rel)/3 or Q = (A+R)/2.

If test data is available, for example if the model is obtained from the output port of a 'Tester' operator in the DesignStudio, A and R are calculated on the test data. If no test data are available, for example if the model is obtained from the output port of a 'Predictor' operator in the DesignStudio, A and R are calculated on the training data, and in this case their values should be regarded with caution because the model might be overfitted and its quality indicators on the training data might not correctly represent the model's predictive power on unknown data.

Class label statistics and co-occurrence counts

- Univariate statistics of the target field: count(t_k) for each target value t_k.
- Co-occurrence counts count($f_{i(j)} \wedge t_k$) for all target values t_k and all values $f_{i(j)}$ of all independent fields f_i.

Field importance information for active fields

Field importance is a value in the range [0…1] indicating the fraction of the model's total predictive power that is contributed by the current field. The sum of all fields' importances equals 1. Importance factors are calculated in two steps:

- First, for each independent field f_i, each value $f_{i(j)}$ of this field and each target field value t_k, the actual co-occurrence number $count(f_{i(j)} \wedge t_k)$ is compared to the expected number $count(f_{i(j)}) \bullet p(t_k)$ that we would have obtained if $f_{i(j)}$ and t_k were statistically independent. For each f_i, we thereby calculate a 'raw' importance number as $\sum_{j=1...N} |count(f_{i(j)}.t_k) - count(f_{i(j)})\ p(t_k)|=5$, where N is the number of different values of field f_i and $|x|\geq 5$ means 'if $|x|\geq 5$ then $|x|$ else 0'. The reason for excluding all summands smaller than 5 is to reduce the impact of random fluctuations that are statistically not significant.
- Second, all 'raw' importance numbers are rescaled so that the sum of all field importances equals 1.

Field-field correlations

The Naïve Bayes Classification model contains field-field correlation coefficients for a subset of all possible field-field pairs (f,g) between active fields f and g.

- If f and g are continuous numeric, DB2 Warehouse calculates linear correlation coefficients:

  $corr(f,g) := \sum_{i=1...N}(f_i-\mu_f)(g_i-\mu_g) / N\sigma_f\sigma_g$

  where (μ_f,σ_f) and (μ_g,σ_g) are mean and standard deviation of the values of fields f and g, and N is the number of data records.
- If f and g are discrete, that means discrete numeric or categorical, DB2 Warehouse calculates adjusted contingency coefficients:

  $cont(f,g) := \sqrt{(k/(k-1) \bullet \chi^2 / (N+ \chi^2))}$

  where χ^2 is the result of the χ^2 test that tests the hypothesis that the value of f significantly influences the value distribution of g (or vice versa). If f has k_f different values and g has k_g different values, then k is defined as $k := min(k_f,k_g)$.
- No correlation is calculated between continuous and discrete fields.
- Furthermore, no correlations are calculated for fields that the mining kernel has removed from the list of active fields because they are either (almost) single valued or (almost) key-like.

Gains charts

A gains chart (or lift chart) is an algorithm-independent visualization and comparison method that allows you to measure and compare the abilities of classification models to reliably separate the data records with high probability of having a certain class label from those ones for which the independent field values indicate a low probability of having that class label.

A Naïve Bayes Classification model contains a separate gains chart for each target field value (class label). A detailed explanation of gains charts can be found in “Gains Chart Extractor operator” on page 380.

Confusion matrix

The confusion matrix, depicted in Figure A-16, is an overview of the predictive accuracy of the model. It compares actual and predicted class labels and shows which misclassifications the model has made on the training or test data.

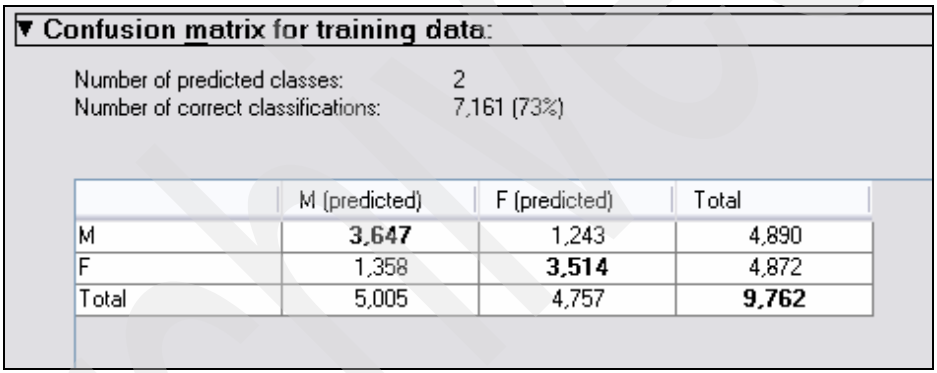


Figure A-16 Confusion matrix

Methods and operators

In this section, we discuss methods and operators for extracting model content into DB2.

For DB2 Warehouse Naïve Bayes Classification models, the Design Studio offers the same operators and API methods for extracting the model content as for Tree Classification models. We therefore refer to “Methods and operators” on page 379. There is only one exception: the operator 'Tree Rules Extractor' and the corresponding SQL API method DM_getGetTreeRules are not available for Naïve Bayes Classification.

Applications of Naïve Bayes Classification

Due to the simplicity of its general approach, Naïve Bayes is a fast and robust classification method that can be applied to almost any classification use case. However, for predominantly continuous numeric data, other classification methods such as Tree Classification might create better models because they do not have to discretize these fields in a preprocessing step.

Naïve Bayes is the method of choice for data with many categorical values, and in particular for text classification scenarios. In text classification, the independent field values are categorical concepts, keywords, or other text characteristics, and the total number of categorical values (keywords) can become very large. Other methods such as Tree Classification often perform poorly on these types of data, whereas Naïve Bayes still runs fast and creates good models.

Further Reading

- http://en.wikipedia.org/wiki/Category:Classification_algorithms
- "An empirical study of the naive Bayes classifier", found at:

  http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf
- "Idiot's Bayes - not so stupid after all?" by D.J. Hand and K. in Yu International Statistical Review. Vol 69 part 3, pages 385-399, (2001).
- "A Comparison of Event Models for Naive Bayes Text Classification", found at:

  http://www.kamalnigam.com/papers/multinomial-aaaiws98.pdf

Logistic Regression based classification

In this section, we discuss Logistic Regression based classification.

Algorithm

Logistic Regression is a widely used classification method for binary classification problems. Imagine, for example, we want to predict the gender of a person from his or her body height. Imagine we have the data of 900 persons and we put them into nine groups of 100 people each, sorted by increasing body height. If we then trace the fraction of males, p('m'), of each group against the group's average body height in meters, we might obtain the (red) points depicted in Figure A-17. These points can be nicely fitted by a sigmoidal curve of the form $p('m') = 1 / (1 + \exp(b_0+b_1 \bullet f))$, where f is the independent field, in our example, body height.

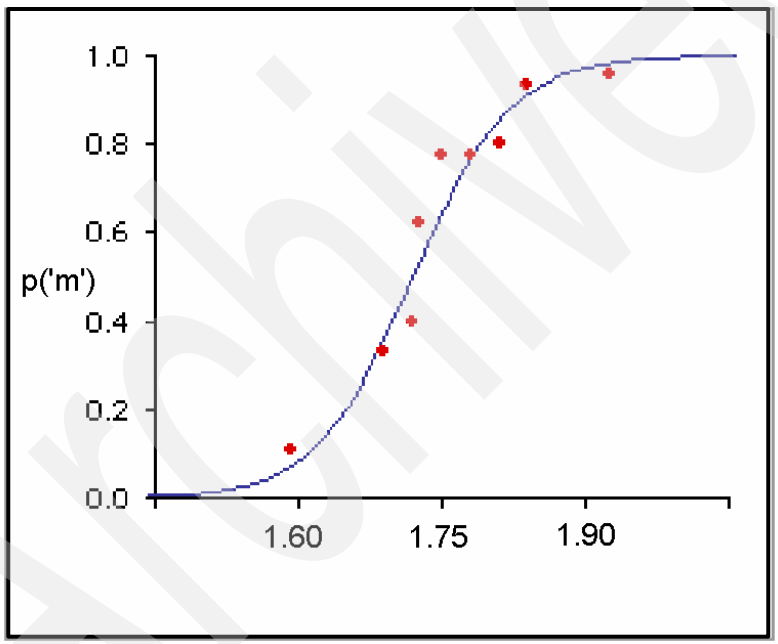


Figure A-17 Logistics Regression

If more than one independent field exists, this formula can be generalized to:

$p = 1 / (1 + \exp(b_0 + \sum_{i=1...n} b_i \bullet f_i))$

When defining logit(p) := log(p / (1-p)), the above formula becomes linear in logit(p):

$logit(p) = b_0 + \sum_{i=1...n} b \bullet f_i$

In other words, the Logistic Regression technique can be classified as a generalized Linear Regression with the logit() function as link function.

In its basic form described so far, Logistic Regression can only work with purely numeric independent fields f_i. Categorical fields must be preprocessed using the so-called *nominal encoding* technique. Nominal encoding replaces one categorical field that has m different values on the training data by m binary numeric fields, which can have the values 0 or 1. In each preprocessed training data record, exactly one of the m binary fields has a value of 1; all others have a value of 0.

The parameter vector $b := (b_0,b_1,\ldots,b_n)$ can be estimated by means of the Maximum Likelihood method. The basic idea of this method is to find those values b that maximize the probability to find the actual target values t_j for all N training data records $(t_j,f_{1j},\ldots,f_{nj})$, where j=1…N. This probability is called the likelihood L:

$$L := \text{proba}((t_1,...,t_N) \mid p(f_{11},\ldots,f_{nN}, b_0,\ldots,b_n)) = \Pi_{j=1..N}\, p(f_j,b)^{tj}\ \ \Pi_{j=1..N}\, (1-p(f_j,b))^{1-tj}$$

For practical computation, it is more convenient to search the maximum of the log-likelihood:

$$\log L := \log(L) = \sum_{j=1..N} (t_j \log(p(f_j,b)) + (1-t_j) \log(1-p(f_j,b)))$$

At the maximum of logL, all derivatives with respect to the b_i must be zero:

$$\delta \log L / \delta b_i = 0$$

These conditions give us a system of n-1 equations that we have to solve for obtaining the best values $b^{(opt)} = (b_0^{(opt)}, \ldots, b_n^{(opt)})$. However, since p is not a linear function but an exponential, a direct solution of this system is not possible. One therefore uses the iterative Newton-Raphson method that starts from a random vector $b^{(0)}$ and subsequently creates improved solutions $b^{(k+1)}$ using the iteration formula:

$$b^{(k+1)} = b^{(k)} - (\delta^2 \log L(b^{(k)}) / \delta b_i \delta b_j)_{ij}^{-1} (\delta \log L(b^{(k)}) / \delta b_i) = b^{(k)} - (H^{(k)})^{-1} D^{(k)}$$

where H is the Hessian matrix of second derivatives of logL and D the vector of first derivatives.

Parameters and advanced options

In the DesignStudio, Logistic Regression based classification is called through the 'Predictor' operator. The Predictor operator in its default state is algorithm agnostic. Once you have defined the target field in the MiningSettings window of the operator's Properties view, the operator either becomes a classifier (if the target field is categorical) or a numeric predictor (if the target field is numeric).

After selection of a categorical target field, the algorithm Logistic Regression can be selected in the drop-down menu called Algorithm. The algorithm supports the following advanced options.

Field Usage Types

In the Column Properties page of the Predictor's Properties view, each available input data field has a default usage type System determined. That means the classification kernel is free to decide which fields it will use as active fields, and which ones are not useful because they are either (almost) single-valued or (almost) key-like or highly correlated with another field. Inactive fields are not used and not even read during the mining; the mining kernel completely neglects the corresponding input data columns.

Modifying the column Field Usage Type in the Column Properties window is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setFldUsageType('fldName',x). If the first argument 'fldName' is missing, the command alters the default usage type for all fields: DM_setFldUsageType(x). Allowed values for x are 1 (active) or 2 (inactive).

Field Types

The Column Properties view of the Predictor Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields, whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one categorical, that means textual, value, and no ordering such as 'x < y' is assumed between different values. Even though the Logistic Regression operator supports this reinterpretation of a numeric field, it is in most cases highly unfavorable to actually do this. As Logistic Regression internally represents each categorical fields with N different values by N binary fields, runtimes and memory consumption is likely to increase considerably.

The following advanced options do not have a directly corresponding input field in the ‘Predictor’ operator properties view. You have to type the API command into the general purpose Optional Parameters field on the Mining Settings window. Figure A-18 shows this for the parameter WhereClause.

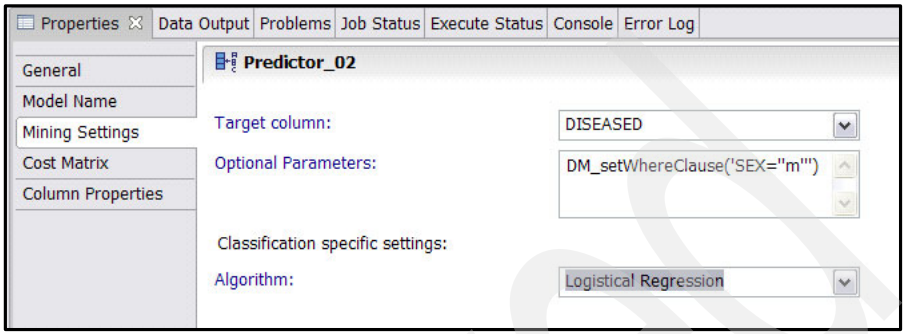


Figure A-18 Operational parameters

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within 'clause' must be surrounded by two single quotes. For Example: DM_setWhereClause('SEX="m"').

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field 'fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

Logistic Regression based classification supports all power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the classification kernel.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.
- -IDM_MAX_DISCR_COUNT and its variants specify how many different valid values a discrete field, for example a categorical field, can have. This power option can be important for Logistic Regression. The default value is 100, which means for each categorical field, the first 100 different values found in the training data are regarded as valid values, while all other values are regarded as invalid. Therefore, one single categorical field can be internally represented by up to 100 binary 0/1 fields, and reducing the number of valid

values to a smaller number can significantly reduce runtimes and memory requirements.

Content of a Logistic Regression Model

DB2 Warehouse Logistic Regression models are stored as PMML <RegressionModel> in the repository table IDMMX.CLASSIFMODELS. They contain the following information.

Model quality

Each Logistic Regression model contains a model quality number Q in the range [0…1]. Q is calculated from two or three detailed quality measures A, R, and Rel:

- Accuracy A = nbCorrectClassifications / nbTrainingOrTestRecords.
- Ranking Quality R = The area between the model's gains chart and a random model's gains chart divided by the area between the best possible gains chart and a random model's gains chart.
- Reliability Rel := A(calculated on training data) / A(calculated on test data). The reliability measure can only be calculated if test data with known class label are available that have not been used to train the model.
- Q is the arithmetical mean of the available detailed quality numbers:

  Model quality Q = (A+R+Rel)/3 or Q = (A+R)/2.

If test data is available, for example, if the model is obtained from the output port of a Tester operator in the DesignStudio, A and R are calculated on the test data. If no test data is available, for example if the model is obtained from the output port of a Predictor operator in the DesignStudio, A and R are calculated on the training data, and in this case their values should be regarded with caution because the model might be overfitted and its quality indicators on the training data might not correctly represent the model predictive power on unknown data.

Field importance information for active fields

Field importance is a value in the range [0…1] indicating the fraction of the model's total predictive power that is contributed by the current field. The sum of all fields' importances equals 1. Importance factors are calculated in two steps:

- First, for each independent field f_i a “raw” importance factor I is calculated as:

  $I = | b_i \bullet \sigma_i |$.

  Here, σ_i is the standard deviation of the N field values f_{ij} with j=1..N. The idea behind this formula is that a field f_i is the more capable of influencing the value of p through the exponential term $\exp(b_i \bullet f_i)$, the higher the average absolute value of $b_i \bullet (f_i - \langle f_i \rangle)$ is, where $\langle f_i \rangle$ is the mean value of f_i.

- Second, all “raw” importance numbers are rescaled so that the sum of all field importances equals 1.

Field-field correlations

The Logistic Regression model contains field-field correlation coefficients for a subset of all possible field-field pairs (f,g) between active fields f and g.

- If f and g are continuous numeric, DB2 Warehouse calculates linear correlation coefficients:

  $corr(f,g) := \sum_{i=1...N}(f_i-\mu_f)(g_i-\mu_g) / N\sigma_f\sigma_g$

  where (μ_f,σ_f) and (μ_g,σ_g) are mean and standard deviation of the values of fields f and g, and N is the number of data records.

- If f and g are discrete, that means discrete numeric or categorical, DB2 Warehouse calculates adjusted contingency coefficients:

  $cont(f,g) := \sqrt{(k/(k-1) \bullet \chi^2 / (N+ \chi^2))}$

  where χ^2 is the result of the χ^2 test that tests the hypothesis that the value of f significantly influences the value distribution of g (or vice versa). If f has k_f different values and g has k_g different values, then k is defined as $k := \min(k_f,k_g)$.

- No correlation is calculated between continuous and discrete fields.

- Furthermore, no correlations are calculated for fields that the mining kernel has removed from the list of active fields because they are either (almost) single valued or (almost) key-like.

Gains charts

A gains chart (or lift chart) is an algorithm-independent visualization and comparison method that allows you to measure and compare the abilities of classification models to reliably separate the data records with high probability of having a certain class label from those ones for which the independent field values indicate a low probability of having that class label.

A Logistic Regression model contains one single gains chart for one of the two target values. A detailed explanation of gains charts can be found in “Gains charts” on page 377.

Confusion matrix

The confusion matrix, depicted in Figure A-19, is an overview of the predictive accuracy of the model. It compares actual and predicted class labels and shows which misclassifications the model has made on the training or test data.

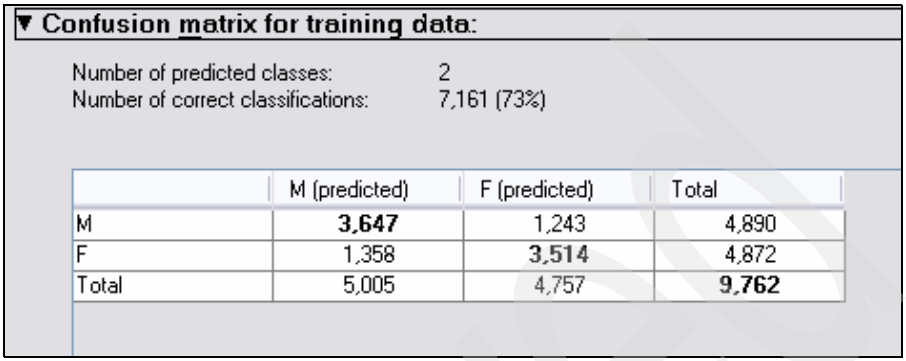


Figure A-19 Confusion matrix

Methods and operators

In this section, we discuss methods and operators for extracting model content into DB2.

For Logistic Regression based classification models, the Design Studio offers the same operators and API methods for extracting the model content as for Tree Classification models. We therefore refer to t“Methods and operators” on page 379. There is only one exception: the operator 'Tree Rules Extractor' and the corresponding SQL API method DM_getGetTreeRules are not available for Logistic Regression.

Applications of Logistic Regression based classification

Logistic Regression is a fast and robust classification method for binary problems. The method is widely applied in social sciences and medicine, for example, when linking the risk of developing a certain disease or medical symptom to other medical, lifestyle, or demographic patient data.

The method is often less reliable when the distribution of the two target values in the training data is very skewed, as it is the case in fraud or deviation detection scenarios. Furthermore, as a generalized Linear Regression technique, the method is unable to correctly detect and represent non-linear, for example, product-like influence factors. If, for example, only male patients with a certain genomic defect suffer from a disease, but the males without the genomic defect or females with or without the defect do not, then a logistic regression model with GENDER and GENOMIC_DEFECT as independent fields is unable to reliably predict a patient's risk to suffer from the disease.

Further reading

- http://en.wikipedia.org/wiki/Logistic_regression
- Hosmer, et al, *Applied Logistic Regression, 2nd Ed.*, Wiley, 2000, ISBN 0471356328.

Scoring a classification model

Once a classification model has been created and visually checked, you might want to apply it to new data for which the target field value is not yet known. Applying a classification model to a table means predicting for each record the target field value, which is also called the class label. In addition to the predicted value you can get other scoring results, for example, the confidence of the prediction, or a list of confidences for all possible class labels.

For this purpose, the Design Studio provides the Scorer operator. This operator takes two inputs, the model and the table to which the model is to be applied. In Figure A-20 on page 400, we have connected a 'Scorer' operator to a classification model that predicts the mortality risk of patients suffering from coronary heart disease. The model calculates the prediction based on patients' demographic data, such as age or sex, and on medical data, such as blood pressure or heart rate. The table matches with the model and provides this required information.

In the Properties view of the Scorer operator, you can select which scoring results will be generated. By default, for each data record the predicted class label and the confidence of this prediction are activated. You will find these columns activated in the Result Items window of the Scorer's Properties view. The exact formula by which the confidence is calculated depends on the classification algorithm that has been used to created the classification model. We will explain the exact formulas for Tree Classification, Naïve Bayes Classification, and Logistic Regression later in this section.

Now we have to provide a table into which the scoring results will be written. By default, the scorer operator tries to write all columns of the input table plus the activated scoring result measures into the result table. If you do not want to reproduce the input table content in the result, you can deactivate certain output columns in the Output Columns window of the Scorer Properties view. In Figure A-20, we are removing all input columns except the primary key column RECORD_ID from the result table.

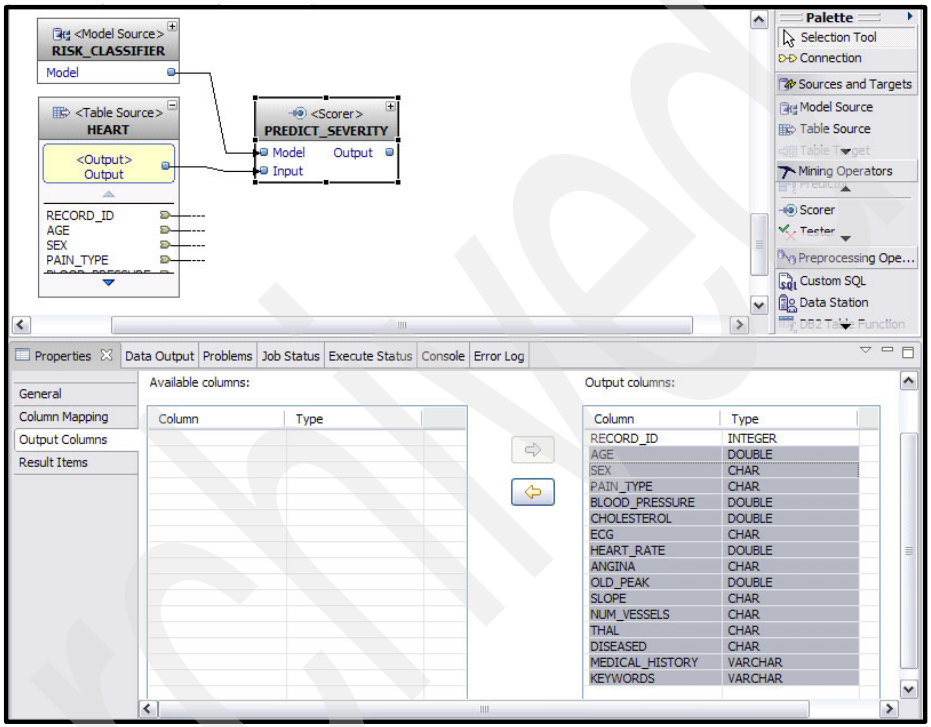


Figure A-20 Properties View of Scorer Operator

In many cases, a result table of an exactly matching column format does not yet exist. Instead of manually creating a new table, you can right-click the output port of the Scorer operator and select the menu item **Create suitable table**. A wizard pops up in which you are presented with a set of default settings for the new table. You can modify these default settings. In Figure A-21, we have marked the column RECORD_ID of the new table as primary key. All other settings are the default settings.

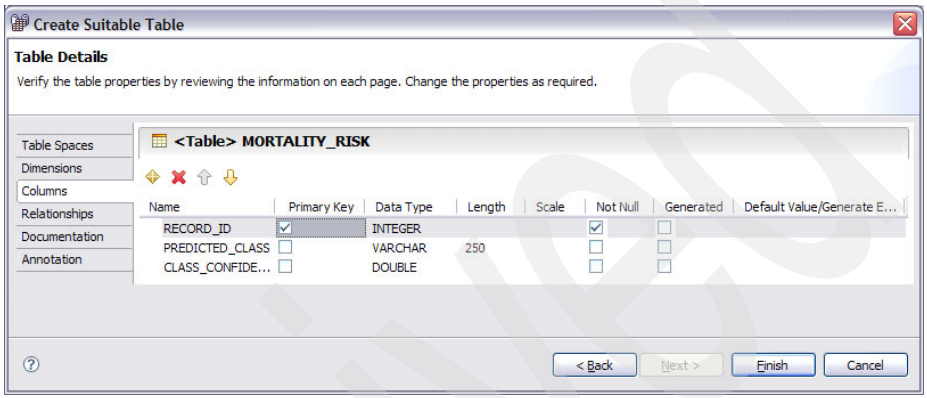


Figure A-21 Table settings

Now we can click the **Finish** button of the table creation wizard, and the resulting mining flow has the desired form, as depicted in Figure A-22. We can execute this flow and fill the table MORTALITY_RISK with the desired scoring results.

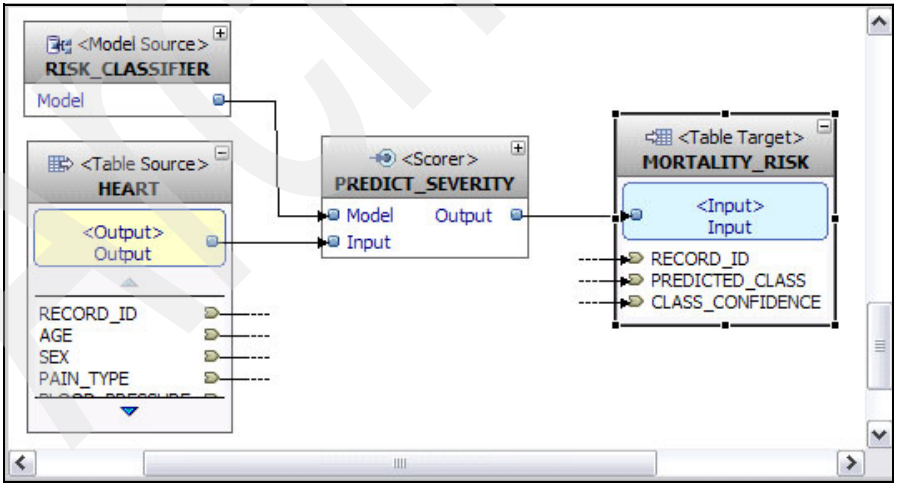


Figure A-22 Mortality risk scoring results

Basic scoring result measures

The graphical panels of the Scorer operator allow modifying and activating the most commonly used result measures of classification scoring.

Predicted Class

This is the predicted class label. It is one of the target field values that existed in the training data on which the model was trained.

Class Confidence

The confidence measure, a probability value between 0 and 1, indicates the certitude of the prediction. From the DesignStudio, only the confidence of the actually predicted value can be activated. Using the DB2 Warehouse SQL API and a custom SQL operator, the confidences for all other class labels can also be retrieved.

For Tree Classification models, confidences are derived from the target value distribution statistics of the tree's leaf node into which the current record falls.

For example, let us assume the current record contains the column values AGE=28 and NB_SERVICES=2, and the model contains a leaf node carrying the condition 'if AGE<30 AND NB_SERVICES<3'. If 150 training data records fell into this leaf node, and 90 of them contained the class label 'STAY', 60 contained the class label 'LEAVE', then the prediction for the current data record will be 'STAY', and the confidence will be 90/150 = 0.6. More precisely, this prediction will be made if no special cost matrix has been defined. Now we assume that the mining flow that created the model contained a cost matrix specifying that predicting 'LEAVE' for a customer who actually STAYs is only half as severe as predicting 'STAY' for a customer who actually LEAVEs. In this case, the model predicts 'LEAVE' with a confidence of 0.4 because committing the less severe misclassification with a probability of 60% causes less damage here than committing the more severe misclassification with a probability of 40%.

In Naïve Bayes Classification, the confidence conf(t) of each possible class label t is computed in two steps. Let n be the number input fields, count the number of training data records, count(t) the number of records with class label t, and count($f_i \wedge t$) the number of records whose i-th input field has the value f_i and whose class label is t. In a first step, a default confidence $conf_0(t) := count(t)/count$ is calculated. This is the probability to find class label t if we do not know anything about the current data record. In a second step, $conf_0(t)$ is multiplied with n increasing or decreasing factors $(count(f_i \wedge t)/count(f_i))/conf_0(t)$. Each of these factors expresses how much the probability to find class label t changes if we know that the input field i has the value f_i. In summary, we get the following formula:

$$conf(t) := conf_0(t) \quad P_{i=1..n} \, (count(f_i.t)/count(f_i))/conf_0(t)$$

In Logistic Regression, the confidence conf(t) of the two possible class labels t_0 and t_1 is computed straightforward from the algorithm's central logistic formula:

$$conf(t_1) := 1 / (1 + \exp(b_0 + \sum_{i=1...n} b_i \bullet f_i))$$

$$conf(t_0) := 1 - conf(t_1)$$

The coefficients $b_0, ..., b_n$ are stored in the Logistic Regression model, and the field values f_i come from the current data record.

Advanced scoring result measures

Using Custom SQL and the DB2 Warehouse SQL Mining API, you can access more advanced scoring results in addition to the ones described above. The Mining API defines a DB2 user defined type (UDT) called DM_ClasResultSpec, which provides a method for retrieving the predicted confidence for all class labels or for one fixed class label:

DM_getConfidence(classLabel)

It returns the predicted confidence (probability) that the current record has the class label classLabel.

The basic result measures described in the preceding section are accessible in the API through:

- DM_getPredClass
- DM_getConfidence

In Example A-2, we show an SQL code that could be embedded into a Design Studio workflow by means of the Custom SQL operator. The code calculates for each data record, which represents a customer, the confidence that the customer will respond positively to a target marketing campaign.

Example: A-2 Custom SQL operator

```
WITH "RESULTSPECIFICATION"( "RESULTSPEC" ) AS
 ( VALUES( IDMMX.DM_ClasResultspec()..DM_setClass('yes') )),
  "CLASSIFVIEW"( "CUSTOMER_ID", "CLASSIFRESULT" ) AS (
    SELECT "CUSTOMER_ID",
    IDMMX.DM_applyClasModel (
      'PredictResponse',
      rec2xml( 1.0, 'COLATTVAL', '',
       B."AGE",B."GENDER",B."MARITAL_STATUS",B."PROFESSION",
       B."INCOME",B."ZIP_CODE" ),
      R."RESULTSPEC"
    )
    FROM "CUSTOMERS" B, "RESULTSPECIFICATION" R
  )
 SELECT
  "CUSTOMER_ID",
  CAST( IDMMX.DM_getConfidence("CLASSIFRESULT",'yes')
    AS DEC(5,4)) AS "PROBA_TO_PURCHASE"
 FROM "CLASSIFVIEW";
```

The three commands shown are three commands of the Mining API. The first one, DM_setClass('yes'), specifies that we want to obtain the confidence of class label 'yes' for each data record. The second command, DM_applyClasModel, reads one data record and calculates its scoring results. The content of this record is provided through the blue rec2xml statement. The third API command, DM_getConfidence(…,'yes'), retrieves the confidence that the current customer will react positively to the target offering.

Regression

A regression model predicts the value of a numeric data field, the target field *t*, in a given data record from the known values of other data fields of the same record, the so-called input fields or explanatory fields or independent fields f_1 to f_n. The independent fields can be numeric or categorical. The predicted value might not be identical to any value contained in the data used to build the model. A regression model is created and trained based on known data sets of data records whose target field values are known. You can apply the trained model to known or to unknown data. In unknown data, the values of the input fields are

known; however, the value of the target field is not known. Applying the model to unknown data is called *scoring*. Applying the model to data with known target values is called *testing*. The purpose of testing is to verify the predictive power and accuracy of the model by comparing the predicted to the actual target values.

Mathematically speaking, a regression model is an equation:

$t = fct(f_1,\ldots,f_n)$

where fct stands for an arbitrary functional dependency. The various regression algorithms and methods differ in the classes of functional dependencies they are able to express and in how these dependencies are established, for example, through direct calculation or through iterative approximation and refinement. DB2 Warehouse offers the following regression algorithms:

- Linear Regression
- Polynomial Regression
- Transform Regression.

Quality measures for regression models

There are many competing methods and algorithms for solving a regression use case. Therefore, it is important to have measures that allow comparing the predictive power of different models created by different algorithms in an objective and unbiased way.

All quality measures described below can be calculated both on the training data during the creation of the model, or on separate test or validation data that were not used to train the model. The latter is the more reliable variant because quality numbers calculated on the training data can be too optimistic if the model is overtrained, meaning that the model predicts the known training data much better than the unknown new data. This may occur if the training set is significantly different from the population and by chance conforms better to the fitting hypothesis.

Even though all DB2 Warehouse regression algorithms contain mechanisms for preventing overtraining, it can never be excluded that the resulting model's quality numbers are slightly too high when calculated on the training data. Therefore, it is always preferable to test a newly created model's predictive power on unknown data before deploying the new model.

Figure A-23 shows how this can be done using the DesignStudio. Here, we want to create a regression model that predicts the number of insurance claims per year that can be expected from an insurance customer with known demographic data. For testing the predictive power of the resulting model, we have put aside 20% of the available historic data. This data is not used during model creation but only afterwards, to test the quality of the model. The ratio of 80% training data and 20% test data is often a good compromise between the goal of using as much as possible of the available data for creating the model and the competing goal of preserving a representative test data set that has not been used during model training.

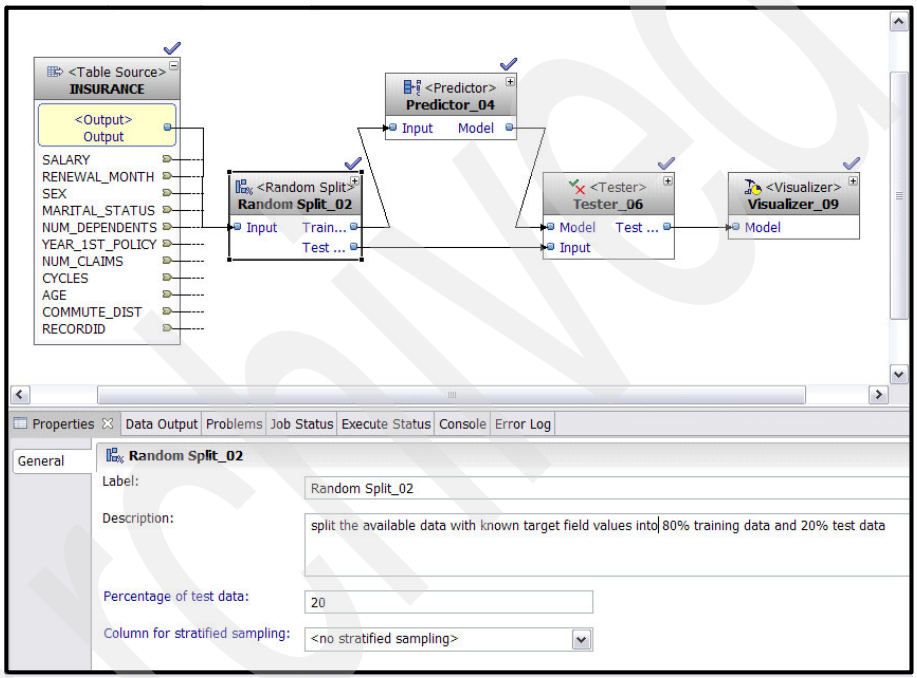


Figure A-23 Regression model

We use the following conventions and notations for defining and explaining the various quality measures:

- There are N training or test data records that are used for quality calculation.
- For each data record i=1...N, we know the actual and the predicted target value, t_i and $t_i^{(pred)}$, and the values of all n independent fields, f_{1i}, …, f_{ni}.
- $\langle t \rangle$ and σ_t are mean and standard deviation of the N target field values, and f_i and σ_j are mean and standard deviation of the j-th independent field f_j.

RMSE (root mean squared error)

As the name says, RMSE is the square root of the average squared prediction error:

$RMSE = (\sum_{i=1...N} (t_i - t_i^{(pred)})^2/N)^{1/2}$.

Pearson's r-Squared

The squared Pearson's correlations coefficient denotes the portion of total variance in the target field that is explained by the regression model:

$r^2 = 1 - \sum_{i=1...N} (t_i^{(pred)} - \langle t \rangle) \bullet / \sum_{i=1...N} (ti - \langle t \rangle)^2$

r^2 does not contain any information about the average accuracy of the single predictions. However, a value significantly above 0 indicates that either the model does not capture an important part of the underlying dependency rules between the independent fields and the target field, or it indicates that the noise level in the data is too high to reliably predict target field values.

A value significantly below 0 indicates that the model contains systematic prediction errors.

Accuracy

The Accuracy quality A measures whether or not the model has systematic errors in its predictions for certain value ranges of the target field. The Accuracy quality captures this by applying a Durbin Watson Test for autocorrelation. For that purpose, the N available training or test data records are ordered by predicted target value $t_i^{(pred)}$ and grouped into a small number of bins, typically M=10…100 bins. Between these bins, A measures the lack of autocorrelation of the average prediction errors:

$A = 1 - |\sum_{i=1...M-1} \langle dt \rangle_i \langle dt \rangle_{i+1} / (\sum_{i=1...M-1} (\langle dt \rangle_i^2 + \langle dt \rangle_{i+1}^2))^{1/2}|$,

where $\langle dt \rangle_i := avg_{bin\ i}(t^{(pred)}-t)$ is the average prediction error in bin i.

If a model has an Accuracy value close to 1, one can be sure that the model does not systematically predict too high or too low values in certain target value ranges. However, a model with high accuracy can still produce high average prediction errors as measured by RMSE. This happens, for example, if the training or test data are very noisy.

Ranking quality

The Ranking quality R measures the area between the model's gains chart and a random model's gains chart, compared to the area between the best possible gains chart and a random model's gains chart.

A gains chart (or lift chart) is an algorithm-independent visualization and comparison method that allows you to measure and compare the abilities of regression models to reliably predict the relative ordering of target field values, which means when two data sets are given, to correctly predict which one has the higher target field value. The gains chart is a two-dimensional plot in which on the x-axis, all N available training or test data records are traced, ordered by decreasing predicted value $t^{(pred)}$. Hence, the x-axis runs from 0 to N. On the y-axis, the gains chart tracks the accumulated sum of target values of all records between x=0 and the current x-axis position. Therefore, each gains chart plot starts at point (0,0) and ends at point $(N,\sum_{i=1...N} t_i)$. The more a model's gains chart plot deviates in an upward direction from the straight line between these two points (the so called random model line), the better is the ranking power of the model.

Figure A-24 depicts the resulting gains chart for the mining flow shown in the introductory part of this section on quality measures. The gains chart has been calculated on the 20% of all available data records that have been reserved for testing the model. These are N=94 records. The blue line in the picture corresponds to the actual model, the dashed green line to the best possible model. The ratio between the shaded area and the area between dashed green and dashed red line is the Ranking quality, which is roughly 0.5 here.

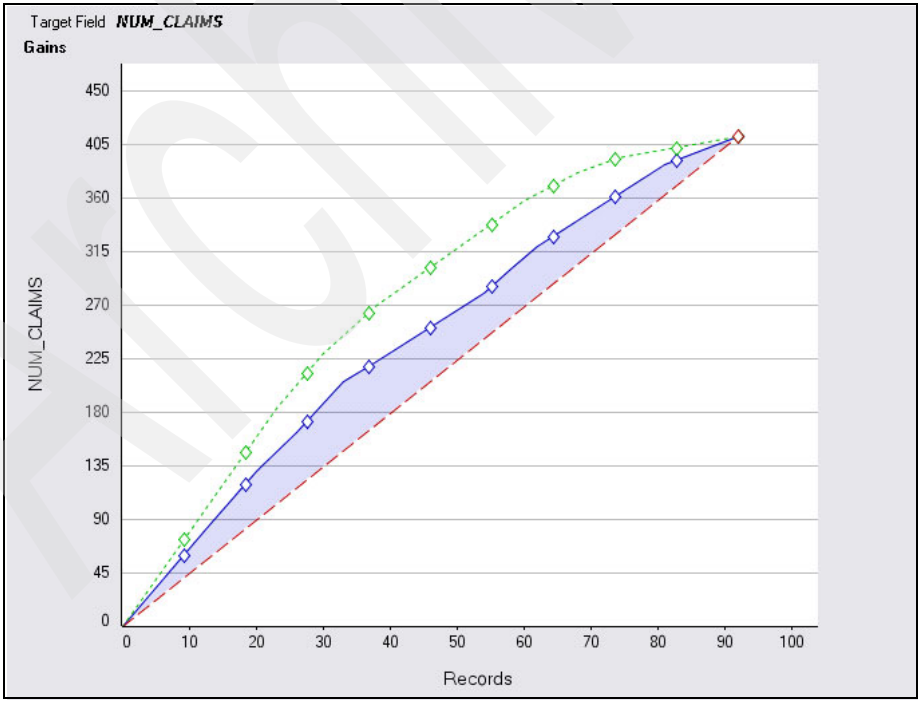


Figure A-24 Gains chart for the mining flow

Reliability

Reliability (Rel) compares the model's RMSE on unknown data to its RMSE on the training data:

$Rel = RMSE_{training} / RMSE_{test}$

The Reliability measure can only be calculated if test data with known class labels are available that have not been used to train the model.

Model quality

Based on the quality measures Accuracy, Ranking Quality, and, if available, Reliability, an overall model quality Q is calculated:

$Q = (A+R+Rel)/3$ or $Q = (A+R)/2$

If test data is available, for example, if the model is obtained from the output port of a *Tester* operator in the DesignStudio, A, R and Rel are calculated on the test data. If no test data is available, for example, if the model is obtained from the output port of a *Predictor operator* in the DesignStudio, A and R are calculated on the training data, and in this case their values should be regarded with caution because the model might be overfitted and its quality indicators on the training data might not correctly represent the model's predictive power on unknown data.

Linear and Polynomial Regression

In this section, we discuss Linear and Polynomial Regression.

Algorithm

The algorithm for Linear Regression assumes a linear relationship between the independent fields.

Linear Regression

The basic Linear Regression algorithm assumes a linear relationship between the n independent fields f_1 to f_n and the target field t. It produces a model that represents a linear equation:

$t = b_0 + b_1 \bullet f_1 + \ldots + b_n \bullet f_n$

The algorithm used in the Linear Regression kernel is an implementation of the least squares method, which minimizes the sum of squared errors of the model:

$|| t - F \bullet b || \rightarrow min$

Here, t is the N-dimensional vector of all records' target values, b is the n-dimensional vector of regression coefficients, and F is the Nxn matrix of all records' independent field values.

If the nxn matrix F^TF is invertible, the (global) minimum can be determined using basic matrix operations:

$b^{opt} = (F^TF)^{-1}F^Tt$

Some fields f_i might be more important than other fields, or some fields might not be important at all. For example, a field might be redundant because it is highly correlated with other fields. Redundant fields can degrade the quality of a model significantly. Therefore, it is essential to apply domain specific knowledge before selecting the explanatory fields or to use the autonomic variable selection. DB2 Warehouse recognizes fields that do not have an explanatory value. To determine whether a field has an explanatory value, the Linear Regression algorithm performs statistical tests in addition to the autonomic variable selection that is performed for all classification and regression methods. If you know the fields that do not have an explanatory value, you can automatically select a subset of the explanatory fields for shorter runtimes by switching the field usage type of certain fields to **inactive** in the Predictor operator Properties view.

The Linear Regression algorithm provides the following methods to automatically select subsets of explanatory fields:

- Stepwise regression: For stepwise regression, you must specify a minimum significance level. Only the fields that have a significance level above the specified value are used by the Linear Regression algorithm.
- r-squared regression: The r-squared regression method identifies an optimum model by optimizing a model quality measure: either the squared Pearson correlation coefficient, or the so-called adjusted squared Pearson correlation coefficient:

  $r^2_{adj} = 1 - (N-1)/(N-p-1) \bullet \sum_{i=1...N} (t_i^{(pred)} - \langle t \rangle)^2 / \sum_{i=1...N} (t_i - \langle t \rangle)^2$.

By default, the Linear Regression algorithm automatically selects subsets of explanatory fields by using the adjusted squared Pearson correlation coefficient to optimize the quality of the model.

In its basic form described so far, Linear Regression can only work with purely numeric independent fields f_i. Categorical fields are internally preprocessed using the so-called nominal encoding technique. Nominal encoding replaces one categorical field that has m different values on the training data by m binary numeric fields that can have the values 0 or 1. In each preprocessed training data record, exactly one out of the m binary fields has a value of 1; all others have a value of 0.

Polynomial Regression

The Polynomial Regression algorithm assumes a polynomial relationship between the independent fields and the target field:

$t = b_0 + b_{11} \bullet f_1 + .. + b_{1d} \bullet f_1^d + \ldots + b_{n1} \bullet f_n + b_{nd} \bullet f_n^d$.

Additionally for the parameters of the Linear Regression algorithm, you can set the maximum degree d of the Polynomial Regression. The default value for the maximum of Polynomial Regression is 3. Before you modify the default value for the maximum degree of Polynomial Regression, take the following considerations into account:

- If you set the maximum degree of Polynomial Regression to 1, the Polynomial Regression algorithm is identical to the Linear Regression algorithm.
- If you specify a high value for the maximum degree of Polynomial Regression, the Polynomial Regression algorithm tends to overfit. This means that the resulting model might approximate the training data very well, but it fails when it is applied to data that is not used for training. Also, a polynomial of high degree tends to behave badly when it is applied to data that is different to a high degree from the input training data.

The Polynomial Regression algorithm uses the following default settings:

- The maximum degree d of the Polynomial Regression is set to 3.
- Subsets of explanatory fields are selected by using the adjusted squared Pearson correlation coefficient to optimize the quality of the model.

Parameters and advanced options

In the DesignStudio, the Regression algorithms are called through the *Predictor operator*. The Predictor operator in its default state is algorithm agnostic. Once you have defined the target field in the MiningSettings window of the operator's Properties view, the operator either becomes a classifier (if the target field is categorical) or a numeric predictor or regressor (if the target field is numeric). After selection of a numeric target field, the algorithms **Linear** and **Polynomial** can be selected in the drop-down menu called Algorithm. The two algorithms support the following advanced options.

Field Usage Types

In the Column Properties page of the Predictor Properties view, each available input data field has a default usage type System determined. That means the classification kernel is free to decide which fields it will use as active fields, and which ones are not useful because they are either (almost) single-valued or (almost) key-like or highly correlated with another field. Only active fields can be used as splitting conditions in tree nodes.

You can overwrite the usage type System determined for any data field if you click the value **System determined** in the table column Field Usage Type. A menu pops up from which you can select either **active** or **inactive**. Active fields will be used for defining the tree's split conditions. Inactive fields are not used and not even read during the mining; the mining kernel completely neglects the corresponding input data columns.

Modifying the column Field Usage Type in the Column Properties window is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setFldUsageType('fldName',x). If the first argument 'fldName' is missing, the command alters the default usage type for all fields: DM_setFldUsageType(x). Allowed values for x are 1 (active) or 2 (inactive).

Field Types

The Column Properties view of the Predictor Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields, whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one categorical, that means textual, value, and no ordering such as $x < y$ is assumed between different values. In the resulting tree model, the field's statistics is stored and visualized in the form of a categorical pie chart, and not as a numeric histogram.

Alternatively, the field type can be set to categorical by typing the following command into the Optional Parameters free text field: DM_setFldType('fldName',0).

The following advanced options do not have a directly corresponding input field in the Predictor operator's properties view. You have to type the algorithm parameter into the Algorithm settings field on the Mining Settings window. Figure A-25 shows this for the parameters Maximum Exponent and Intercept.

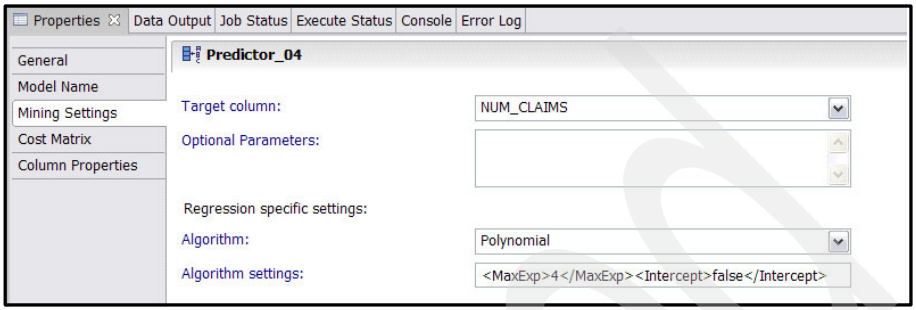


Figure A-25 Parameters Maximum Exponet and Intercept

Maximum Exponent

You can specify the maximum polynomial degree d for Polynomial Regression. The default value for d in Polynomial Regression is 3. In Linear Regression, this parameter is ignored since d is fixed to 1. You can type the following string into the Algorithm settings text box for manually defining the maximum polynomial degree d, where d can be any number between 1 and 7:

```
<MaxExp>d</MaxExp>
```

Intercept

Per default, the regression formula for Linear and Polynomial Regression can contain an additive offset b_0, called intercept. If an additive constant is not desired, b_0 can be disabled by typing the following string into the Algorithm settings'text box:

```
<Intercept>false</Intercept>
```

Minimum Significance Level

Per default, the Linear and Polynomial Regression kernels select the optimum subset of independent fields to be used in the regression by using the adjusted squared Pearson correlation coefficient to optimize the quality of the generated model. You can switch to the so-called Stepwise Regression method in which, iteratively, one independent field a time is added to the regression equation. Each time a new field is to be added, the field with the highest significance level is chosen. The significance level is a number between 0 and 1 calculated from an F-test, which tests that the field and significantly impacts the target field. The iterative field addition step is continued until the best remaining field's significance level drops below the user-defined minimum level. The lower the

threshold is, the more accurate the prediction on the training data, and the higher the risk to produce an overtrained model.

You can type the following string into the Algorithm settings text box for switching to Stepwise Regression and for specifying the minimum significance level 0.3:

```
<MinSigLevel>0.3</MinSigLevel>
```

r-Squared mode

Per default, the Linear and Polynomial Regression kernels select the optimum subset of independent fields to be used in the regression by using the adjusted squared Pearson correlation coefficient to optimize the quality of the generated model. You can switch to using the normal, unadjusted squared Pearson correlation coefficient by typing the following string into the Algorithm settings text box:

```
<RSquaredMode>normal</RSquaredMode>
```

Note: The two parameters <RSquaredMode> and <MinSigLevel> are mutually exclusive. Only one of them should be used.

The following advanced options do not have a directly corresponding input field in the ‘Predictor’ operator's properties view. You have to type the API command into the general purpose Optional Parameters field on the Mining Settings window. Figure A-26 shows this for the parameters Field Outlier Treatment and Where Clause.

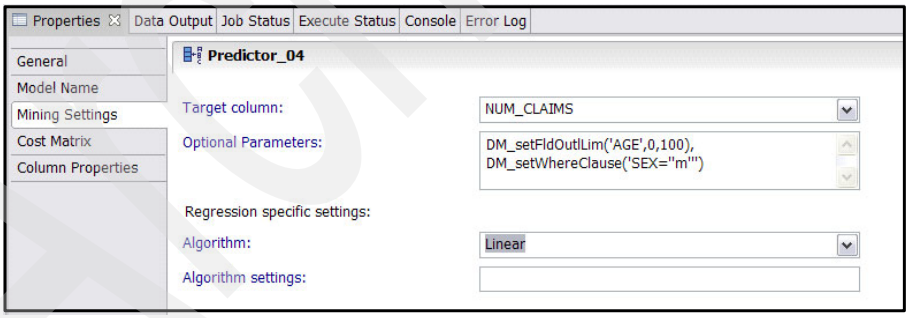


Figure A-26 Optional parameters

Field outlier treatment

For numeric fields, a valid value range can be defined, and once a valid range setting exists, an outlier handling method can be defined that specifies how to treat field values that lie below or above the allowed range. The following outlier treatment methods for values outside the valid range exist:

- 0: 'as is': The original value is kept unchanged. This method is the default.
- 1: 'as missing': The original value is replaced by SQL NULL.
- 2: 'as extreme': The original value is replaced by the upper, respectively lower boundary of the valid range if it lies above, respectively below the valid range.

And here are the two commands that DB2 Warehouse provides for this purpose:

```
DM_setFldOutlLim('fieldname',lower,upper)
DM_setFldOutlTreat('fieldname',methodID)
```

Here, lower and upper can be values of any integer, date or time, or floating point type; the type must fit with the corresponding field type. It is possible to specify only one range boundary; the other argument must then be 'CAST (NULL AS fieldType)'. methodID is either 0, 1, or 2.

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within 'clause' must be surrounded by two single quotes. Here is an example:

```
DM_setWhereClause('GENDER=''m''')
```

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field 'fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

The Linear and Polynomial Regression kernels support all power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the mining kernel.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.
- -IDM_MAX_DISCR_COUNT and its variants specify how many different valid values a discrete field, for example, a categorical field, can have. The default value is 100, which means for each categorical field, the first 100 different values found in the training data are regarded as valid values; all other values

are regarded as invalid. Increasing the threshold to a value larger than 100 can increase the quality of the generated model, but it might slow down the training process.

Content of a Linear or Polynomial Regression models

DB2 Warehouse Linear and Polynomial Regression models are stored as PMML <RegressionModel> in the repository table IDMMX.REGRESSIONMODELS. They contain the following information:

Regression Equation

The most important info in the regression model is the detected regression equation, represented by the coefficients b_0, …, b_n for Linear Regression or b_0, b_{11}, …, b_{nd} for Polynomial Regression.

Model qualities

Each DB2 Warehouse regression model contains the six model quality numbers described above:

- Root mean squared error (RMSE)
- Pearson's r-squared (r^2)
- Accuracy (A)
- Ranking Quality (R)
- Reliability (Rel) (only if test data are available)
- Overall Model Quality (Q)

Field importance information for active fields

For each input data field that has been used as an active field in the regression, the model contains a field importance indicator: a value in the range [0…1] indicating the fraction of the model's total predictive power that is contributed by the current field. The sum of all fields' importances equals to 1. Importance factors are calculated in several steps.

1. First, for each active field, a raw importance number I_{raw} is calculated. For numeric fields f_i, $I_{raw}(i)$ is the absolute slope of the regression equation with respect to f_i after normalization, which means after replacing f_i by $f'_i := (f_i - \langle f_i \rangle)/\sigma_i$:

   $I_{raw}(i) = | \partial t^{(pred)} / \partial f'_i |$

   For categorical fields f_i, $I_{raw}(i)$ is the weighted arithmetical mean of the single field values regression coefficients. If the field has m(i) different values, we get:

   $I_{raw}(i) = (\sum_{k=1\ldots m(i)} N_{ik} \bullet |b_{ik}|) / N,$

   where N_{ik} is the number of data records having the k-th value of field i and N is the total number of data records.

2. Second, all raw importance numbers are rescaled so that the sum of all field importances equals 1.

Field-field correlations

The regression model contains field-field correlation coefficients for a subset of all possible field-field pairs (f,g) between active fields f and g.

- If f and g are continuous numeric, DB2 Warehouse calculates linear correlation coefficients:

  $corr(f,g) := \sum_{i=1...N}(f_i-\mu_f)(g_i-\mu_g) / N\sigma_f\sigma_g$

  where (μ_f,σ_f) and (μ_g,σ_g) are mean and standard deviations of the values of fields f and g, and N is the number of data records.

- If f and g are discrete, that means discrete numeric or categorical, so DB2 Warehouse calculates adjusted contingency coefficients:

  $cont(f,g) := \sqrt{(k/(k-1) \bullet \chi^2 / (N+ \chi^2))}$

  where χ^2 is the result of the χ^2 test that tests the hypothesis that the value of f significantly influences the value distribution of g (or vice versa). If f has k_f different values and g has k_g different values, then k is defined as $k := \min(k_f,k_g)$.

- No correlation is calculated between continuous and discrete fields.

- Furthermore, no correlations are calculated for fields that the mining kernel has removed from the list of active fields because they are either (almost) single valued or (almost) key-like.

Gains charts

Each regression model contains a gains chart as described in the introductory section on quality measures in regression models. The gains chart is a graphical visualization showing to which extent the model is able to predict the relative ordering of the training or test data records. If test data is available, the gains chart inside the DB2 Warehouse regression model is based on this test data; otherwise it is based on the training data. In this second case, the gains chart might be too optimistic if the model is overtrained.

Methods and operators

In this section, we discuss the methods and operators for extracting model content into DB2 tables.

Gains Chart Extractor operator

The mining flow editor of the Design Studio contains an operator 'Gains Chart Extractor' that writes the model's gains chart data into a table containing the following columns:

- THRESHOLD (double): The minimum target value in the current bin. This is the lower bin boundary. For the last bin, which has no lower boundary, SQL NULL is returned.
- ROWCOUNT (bigint): The accumulated number of all test or training data records that have a predicted target value equal to or larger than this bin's THRESHOLD.
- SUMACTUAL (double): The accumulated sum of all actual target values of all test or training records for which the predicted target value is at least THRESHOLD.

Quality Extractor operator

The Quality Extractor, which can also be used with any other DB2 Warehouse mining model, writes four quality numbers into a four-column table:

- MODELQUALITY (real): Overall model quality Q as defined above
- PREDICTIONACCURACY (real): Accuracy A as defined above.
- RANKINGQUALITY (real): Ranking quality R as defined above.
- RELIABILITY (real): Reliability quality Rel as defined above.

SQL API commands and Custom SQL operator

The DB2 Warehouse mining SQL API offers some more model extractor methods for which no specific graphical mining flow operator exists. These commands can be used within a mining flow by means of the Custom SQL operator. Figure A-27 shows how the API command DM_getRMSE is called within a Custom SQL operator. Figure A-27 shows the Properties view of a Custom SQL operator, which has been drawn to the DesignStudio' Mining Flow Editor canvas and whose input port has been connected to the table source RMSE.

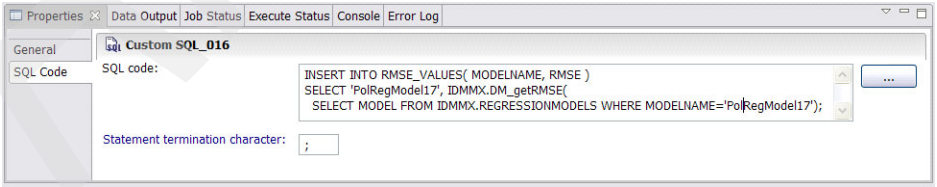


Figure A-27 Custom SQL operator

This operator writes the name and the root mean squared error of the Polynomial Regression model 'PolRegModel17' into the two columns MODELNAME and RMSE of the table RMSE_VALUES.

The following model introspection commands for Linear or Polynomial Regression models, which are stored as objects of type IDMMX.DM_RegModel in the table IDMMX.REGRESSIONMODELS, are available:

- IDMMX.DM_expRegModel(regModel): Returns the PMML model as a flat XML file in the form of a CLOB.
- IDMMX.DM_getRegTarget(regModel): Returns the name of the target field.
- IDMMX.DM_getMdlQuality(regModel): Returns the model quality Q.
- IDMMX.DM_getReliability(regModel): Returns the reliability quality Rel.
- IDMMX.DM_getRankQuality(regModel): Returns the ranking quality R.
- IDMMX.DM_getPredAccuracy(regModel): Returns the accuracy quality A.
- IDMMX.DM_getRegEquations(regModel): Returns the model's regression equations that are needed to calculate the predicted value. The equations are provided as a textual string in the form of a piece of valid C code. You can use the output of DM_getRegEquations for:
  - Auditing purposes: Document how a score was calculated.
  - Understanding the contribution of each input field.
  - Directly embedding the model into a C, C++, or Java application.
- IDMMX.DM_getGainsChart(regModel): This command is the API equivalent of the Gains Extractor operator. It returns a three column table (ROWCOUNT, SUMACTUAL, and THRESHOLD) in which each row describes one point of the gains chart. ROWCOUNT is the x-coordinate of the point, SUMACTUAL the y-coordinate.
- IDMMX.DM_getQualities(regModel): This command is the API equivalent of the Qualities Extractor operator. It returns a two column table (QUALITYNAME and QUALITYVALUE) in which each row contains one of the quality numbers model quality Q, accuracy A, ranking quality R, or reliability Rel.
- IDMMX.DM_getFields(clasModel): Returns a four column table (COLNAME, FIELDNAME, MININGTYPE, and IMPORTANCE) that contains for each active field in the model the name of the data column to which it refers, its alias name, its mining type (0 for categorical or 1 for numeric), and its importance, a value between 0 and 1. The sum of all importance values is 1.

- IDMMX.DM_getCorrelations(clasModel): Returns a three column table (FIELDNAME1, FIELDNAME2, and CORRELATION) that lists all field-field correlation numbers contained in the model. Correlations between continuous numeric fields are linear correlation coefficients, correlations between discrete fields are adjusted contingency coefficients, and correlations between discrete and continuous fields are not calculated.

Applications of Linear or Polynomial Regression

Linear and Polynomial Regression are rather limited in the types of functional dependencies they can model correctly. In particular, they completely ignore mixed factors such as $f_i \bullet f_k$. Therefore, Linear or Polynomial Regression should not be used as default regression algorithms. For many use cases and for many types of input data, the Transform Regression algorithm produces models with higher prediction accuracy.

Linear or Polynomial Regression should be used instead of Transform Regression mainly in two situations:

1. If the input data is so large that the runtime of Transform Regression becomes unacceptably long.
2. If an easily understandable model is to be generated whose regression formula can be easily interpreted. In a Linear Regression model, the regression coefficients for each field directly show in which direction and with which strength each input field influences the prediction. In contrast to this, a Transform Regression model is a complex concatenation of several partial models whose interpretation in terms of easily understandable rules is much more difficult.

Further reading

- http://en.wikipedia.org/wiki/Linear_regression
- http://en.wikipedia.org/wiki/Stepwise_regression
- *Applied Regression Analysis, Linear Models and Related Methods*, by Fox
- Efroymson, “Multiple regression analysis", in *Mathematical Methods for Digital Computers* by Ralston and Wilf (ed.)
- Introduction to contingency tables:

  http://en.wikipedia.org/wiki/Contingency_table

Transform Regression

In this section, we discuss the Transform Regression algorithm.

Algorithm

The Transform Regression algorithm is an IBM patented algorithm. It iteratively combines stepwise Linear Regression and nonlinear field transformations. The idea behind the algorithm is to combine the advantages of Decision Trees with Linear Regression models in the leaves with Neural Networks. In many respects, decision trees and neural networks represent diametrically opposed classes of learning techniques. A strength of one is often a weakness of the other. Decision Tree methods approximate response surfaces of target field values $t(f_1,\ldots,f_n)$ by segmenting the input space of independent fields into regions and using simple models within each region for local surface approximation.

The strengths of Decision Tree methods are that they are nonparametric, fully automated, and computationally efficient. Their weakness is that statistical estimation errors increase with the depth of trees, which ultimately limits the granularity of the surface approximation that can be achieved for fixed sized data. In contrast, Neural Network methods fit highly flexible families of nonlinear parametric functions to entire surfaces to construct global approximations. The strength of this approach is that it avoids the increase in estimation error that accompanies segmentation and local model fitting.

The weakness is that fitting nonlinear parametric functions to data is computationally demanding, and these demands are exacerbated by the fact that often several network architectures need to be trained and evaluated in order to maximize predictive accuracy.

Figure A-28 is a depiction of how the Transform Regression algorithm works.

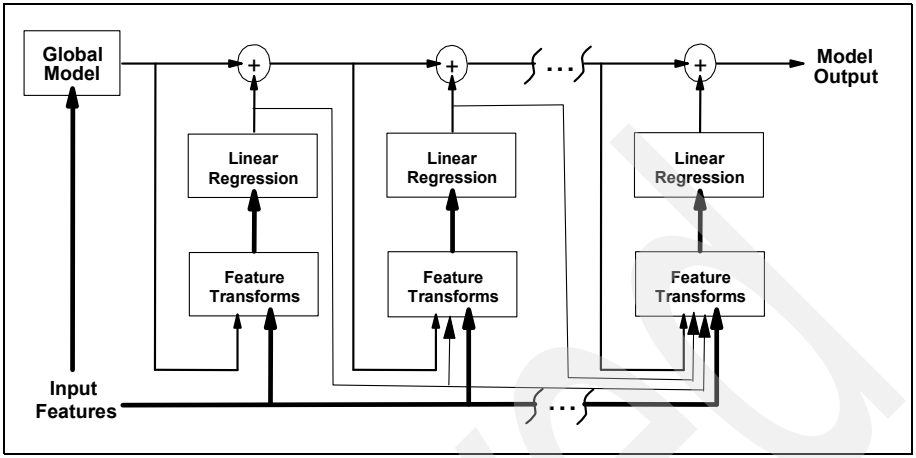


Figure A-28 Transform Regression algorithm

The Transform Regression algorithm works as follows:

- An initial global model is created from the originally available independent fields, which are called input features in the Transform Regression terminology.
- This first model is then iteratively refined by combinations of so-called feature transforms and subsequent Linear Regressions, which use both the initial input features and the transformed features of all previous iteration steps as possible input variables.
- The feature transforms creates decision trees, which are able to handle nonlinearities, as depicted in Figure A-29.

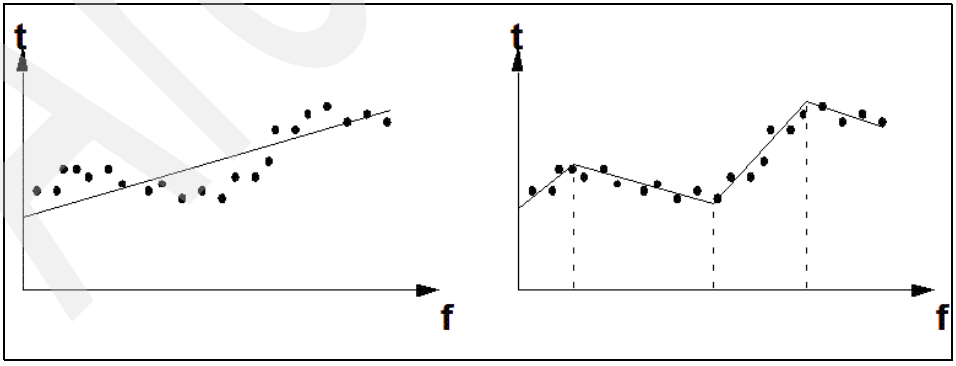


Figure A-29 Decision Trees

- Gradient boosting is used to iteratively improve model accuracy by adding successive output correction stages.
- Cross-product interactions are implicitly handled by using the outputs of previous stages as inputs to subsequent stages. After the first model refinement iteration, the created model implicitly handles all simple cross-product interactions ($f_i \bullet f_k$), after the second refinement all three-factor interactions ($f_i \bullet f_k f_l$), and so on.

When the Transform Regression algorithm reads the training data to create a model, the available data are split into several parts, the so-called data channels, in order to implement a sophisticated cross validation mechanism, as depicted in Figure A-30.

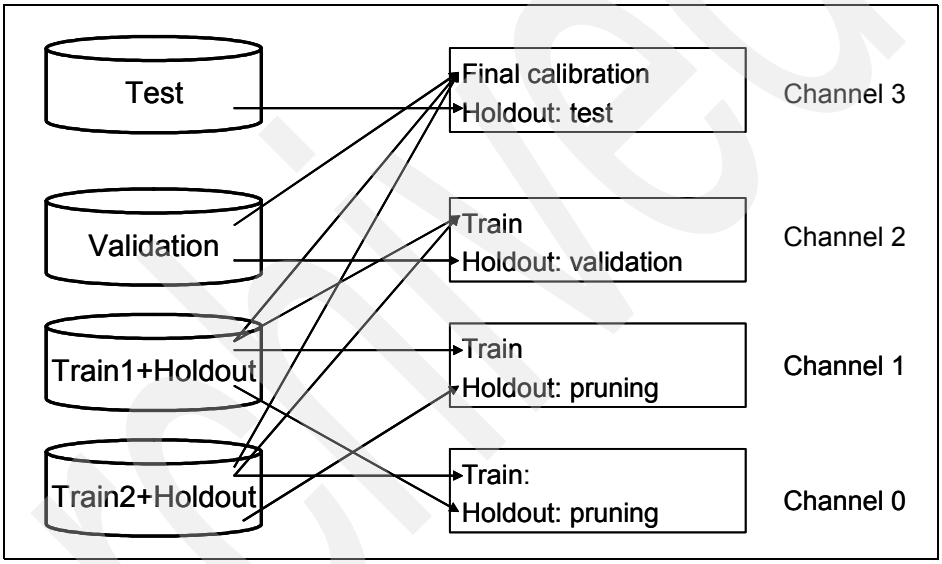


Figure A-30 Data Channels

The main training data is split into two channels. On each channel, a model is trained. The data from the other channel is used to prune the model to prevent overtraining. Optionally, the user can specify a separate validation data set. This set is not used as training data while the model structure is trained and pruned. Instead, it serves to assess the quality of the generated model structure, and it is used to fine-tune the model's coefficients once the structure of the equations has been defined.

Unlike validation data, test data is only used to measure the predictive power of the model on unknown data. Test data does not influence the created model in any way.

Parallel Mode

Transform Regression has a parallel mode, depicted in Figure A-31, in which the algorithm exploits DB2 parallelization features to run in parallel on several CPUs and on several data partitions. In the parallel mode, the DB2 stored procedure in which the main mining process runs issues several parallel DB2 user defined functions (UDFs) that read the training data and the required statistical aggregations that are needed to create the next iteration's refined Transform Regression model. In its final call, each UDF returns this statistical aggregations, and the main stored procedure aggregates all partial results into the next refined model.

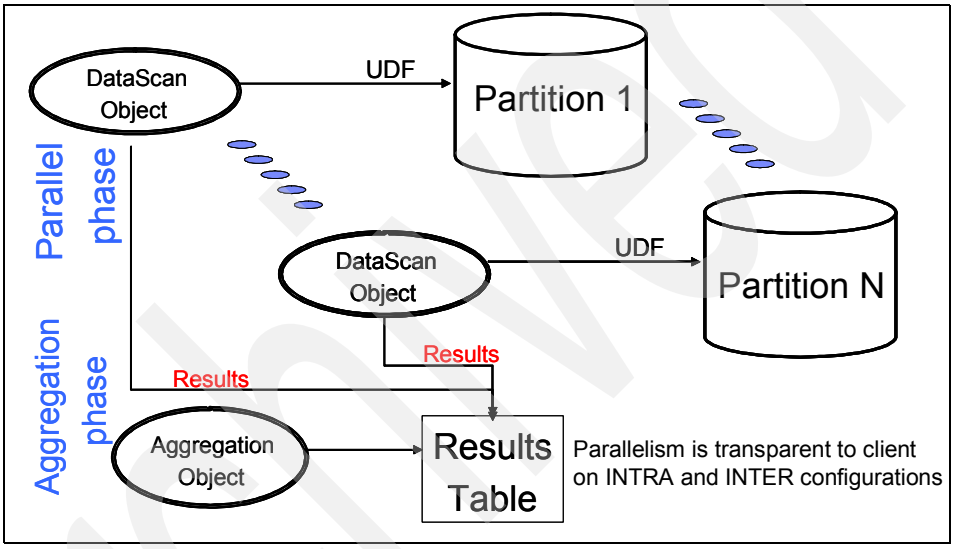


Figure A-31 Parallel Mode

Transform Regression automatically starts either in serial or in parallel mode, depending on the computer system and DB2 configuration you are using. Parallel mode is chosen when the following conditions are satisfied:

- The DB2 configuration parameter INTRA_PARALLEL is set to YES. You can change that parameter by typing the following command in the DB2 command window:

  ```
  db2 update dbm cfg using INTRA_PARALLEL=YES
  ```

- The DB2 configuration parameter MAX_QUERYDEGREE is set to a value larger than 3. You can change that parameter by typing the following command in the DB2 command window (with d being an integer larger than 3):

  ```
  db2 update dbm cfg using MAX_QUERYDEGREE=x
  ```

Parameters and advanced options

In the DesignStudio, the Regression algorithms are called through the Predictor operator. The Predictor operator in its default state is algorithm agnostic. Once you have defined the target field in the MiningSettings window of the operator's Properties view, the operator either becomes a classifier (if the target field is categorical) or a numeric predictor or regressor (if the target field is numeric). After selecting the numeric target field, the algorithm Transform Regression is the default regression algorithm. Transform Regression can also be manually selected in the drop-down menu called Algorithm. The algorithm supports the following advanced options.

Field Usage Types

In the Column Properties page of the Predictor's Properties view, each available input data field has a default usage type of System determined. That means the classification kernel is free to decide which fields it will use as active fields, and which ones are not useful because they are either (almost) single-valued or (almost) key-like or highly correlated with another field. Only active fields can be used as splitting conditions in tree nodes.

You can overwrite the usage type System determined for any data field if you click the value **System determined** in table column Field Usage Type. A menu pops up from which you can select either **active** or **inactive**. Active fields will be used for defining the tree's split conditions. Inactive fields are not used and not even read during the mining; the mining kernel completely neglects the corresponding input data columns.

Modifying the column Field Usage Type in the Column Properties window is a shortcut for and has the same effect as typing the following command into the Optional Parameters free text field: DM_setFldUsageType(fldName,x). If the first argument *fldName* is missing, the command alters the default usage type for all fields: DM_setFldUsageType(x). Allowed values for x are 1 (active) or 2 (inactive).

Field Types

The Column Properties view of the Predictor Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of a numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields, whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one categorical, which means textual, value, and no ordering such as x < y is assumed between different values. In the resulting tree model, the field's statistics is stored and visualized in the form of a categorical pie chart, and not as a numeric histogram.

Alternatively, the field type can be set to categorical by typing the following command into the Optional Parameters free text field: DM_setFldType('fldName',0).

The following advanced options do not have a directly corresponding input field in the Predictor operator Properties view. You have to type the API command into the general purpose Optional Parameters field on the Mining Settings window. Figure A-32 shows this for the parameters Maximum Execution Time and Where Clause.

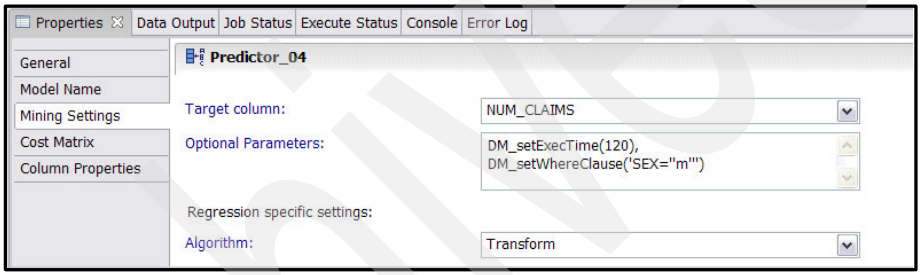


Figure A-32 Optional parameters

Maximum Execution Time

Before starting a Transform Regression modeling run, the user can specify a desired maximum execution time by typing the following command into the Optional Parameters box:

```
DM_setExecTime(x)
```

where x is the desired maximum runtime in minutes. When this option is specified, the algorithm tries to limit its runtime to not more than the desired amount, for example, by sampling the training data, or by terminating before the convergence criterion is reached. The kernel uses some heuristics based on CPU speed and other criteria for predicting the probable runtime. This heuristic is not always perfectly reliable, so the actual runtime might exceed the desired one by a couple of minutes.

Where clause

The command DM_setWhereClause(clause) specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within clause must be surrounded by two single quotes. Here is an example:

```
DM_setWhereClause('GENDER=''m''')
```

Field Alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field fldName. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

Transform Regression supports the following power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the mining kernel.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs. This power option is particularly important for Transform Regression because this algorithm temporarily writes binary data objects containing a compressed representation of the training data into the working directory. On very large data, the default temporary directory might not contain enough free disk space, and the temporary output must be redirected.

Content of a Transform Regression model

Transform Regression models are stored as a PMML <MiningModel algorithmName="transform regression">. The <MiningModel> contains several <Regression> and <DecisionTree> submodels. The models are stored in the repository table IDMMX.REGRESSIONMODELS. They contain the following information.

Model equations

The most important information in the Transform Regression model is the detected tree structures and regression equations.

Model qualities

Each DB2 Warehouse regression model contains the six model quality numbers described above:

- Root mean squared error (RMSE)
- Pearson's r-squared (r^2)
- Accuracy (A)
- Ranking Quality (R)

- Reliability (Rel) (only if test data is available)
- Overall Model Quality (Q)

Field importance information for active fields

For each input data field that has been used as an active field in the regression, the model contains a field importance indicator: a value in the range [0…1] indicating the fraction of the model's total predictive power that is contributed by the current field. The sum of all fields' importances equals to 1. Importance factors are calculated in several steps:

1. First, for each active field f_i a raw importance number I_{raw} is calculated. $I_{raw}(i)$ is the absolute sensitivity of the predicted value with respect to f_i after normalization, which means after replacing f_i by $f'_i := (f_i - \langle f_i \rangle)/\sigma_i$. In Linear and Polynomial Regression, where the model consists of one single equation, we were able to directly calculate this sensitivity as the derivative:

   $I_{raw}(i) = | \partial t^{(pred)} / \partial f'_i |$.

   In Transform Regression with its nested models, this derivative cannot be calculated. Instead, we randomly sample about 100 data records and their f_i values, then slightly perturb these f_i values, obtaining $g_i := f_i + d_f$. After normalization of f_i and g_i, we can calculate for each sample point the approximated derivative as:

   $I_{raw}(i) = | (t^{(pred)}(g_i) - t^{(pred)}(f_i)) / (g'_i - f'_i) |$

2. Second, all raw importance numbers are rescaled so that the sum of all field importances equals 1.

Field-field correlations

The regression model contains field-field correlation coefficients for a subset of all possible field-field pairs (f,g) between active fields f and g.

- If f and g are continuous numeric, DB2 Warehouse calculates linear correlation coefficients:

  $corr(f,g) := \sum_{i=1...N}(f_i-\mu_f)(g_i-\mu_g) / N\sigma_f\sigma_g$

  where (μ_f,σ_f) and (μ_g,σ_g) are mean and standard deviation of the values of fields f and g, and N is the number of data records.

- If f and g are discrete, that means discrete numeric or categorical, DB2 Warehouse calculates adjusted contingency coefficients:

  $cont(f,g) := \sqrt{(k/(k-1) \bullet \chi^2 / (N+ \chi^2))}$

  where χ^2 is the result of the χ^2 test that tests the hypothesis that the value of f significantly influences the value distribution of g (or vice versa). If f has k_f different values and g has k_g different values, then k is defined as $k := \min(k_f,k_g)$.

- No correlation is calculated between continuous and discrete fields.
- Furthermore, no correlations are calculated for fields that the mining kernel has removed from the list of active fields because they are either (almost) single valued or (almost) key-like.

Gains charts

Each Transform Regression model contains a gains chart as described in the introductory section on quality measures in regression models. The gains chart is a graphical visualization showing which extent the model is able to predict the relative ordering of the training or test data records. If test data is available, the gains chart inside the DB2 Warehouse regression model is based on this test data; otherwise it is based on the training data. In this second case, the gains chart might be too optimistic if the model is overtrained.

Methods and operators

In this section, we discuss the methods and operators for extracting model content into DB2 tables.

Gains Chart Extractor operator

The mining flow editor of the Design Studio contains an operator 'Gains Chart Extractor' that writes the model's gains chart data into a table containing the following columns:

- THRESHOLD (double): The minimum target value in the current bin. This is the lower bin boundary. For the last bin, which has no lower boundary, SQL NULL is returned.
- ROWCOUNT (bigint): The accumulated number of all test or training data records that have a predicted target value equal to or larger than this bin's THRESHOLD.
- SUMACTUAL (double): The accumulated sum of all actual target values of all test or training records for which the predicted target value is at least THRESHOLD.

Quality Extractor operator

The Quality Extractor, which can also be used with any other DB2 Warehouse mining model, writes four quality numbers into a four-column table:

- MODELQUALITY (real): Overall model quality Q as defined above
- PREDICTIONACCURACY (real): Accuracy A as defined above.
- RANKINGQUALITY (real): Ranking quality R as defined above.
- RELIABILITY (real): Reliability quality Rel as defined above.

SQL API commands and Custom SQL operator

The DB2 Warehouse mining SQL API offers some more model extractor methods for which no specific graphical mining flow operator exists. These commands can be used within a mining flow by means of the Custom SQL operator. Figure A-33 shows how the API command DM_getRMSE is called within a Custom SQL operator. Figure A-33 shows the Properties view of a Custom SQL operator, which has been drawn to the DesignStudio' Mining Flow Editor canvas and whose input port has been connected to the table source 'RMSE'.

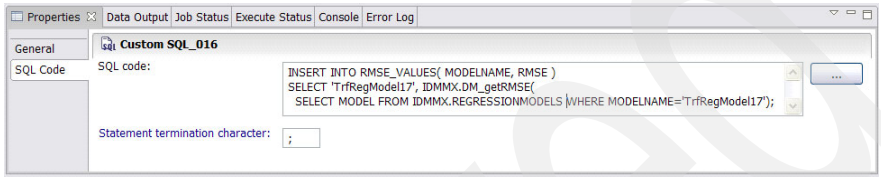


Figure A-33 Custom SQL operator

This operator writes the name and the root mean squared error of the Transform Regression model 'TrfRegModel17' into the two columns MODELNAME and RMSE of the table RMSE_VALUES.

The following model introspection commands are available for Transform Regression models, which are stored as objects of type IDMMX.DM_RegModel in the table IDMMX.REGRESSIONMODELS:

- IDMMX.DM_expRegModel(regModel): Returns the PMML model as a flat XML file in the form of a CLOB.
- IDMMX.DM_getRegTarget(regModel): Returns the name of the target field.
- IDMMX.DM_getMdlQuality(regModel): Returns the model quality Q.
- IDMMX.DM_getReliability(regModel): Returns the reliability quality Rel.
- IDMMX.DM_getRankQuality(regModel): Returns the ranking quality R.
- IDMMX.DM_getPredAccuracy(regModel): Returns the accuracy quality A.
- IDMMX.DM_getRegEquations(regModel): Returns the model's regression equations that are needed to calculate the predicted value. The equations are provided as a textual string in the form of a piece of valid C code. You can use the output of DM_getRegEquations for:
  - Auditing purposes: Document how a score was calculated.
  - Understanding the contribution of each input field.
  - Directly embedding the model into a C, C++, or Java application.

- IDMMX.DM_getGainsChart(regModel): This command is the API equivalent of the Gains Extractor operator. It returns a 3-column table (ROWCOUNT, SUMACTUAL, and THRESHOLD) in which each row describes one point of the gains chart for. ROWCOUNT is the x-coordinate of the point, SUMACTUAL the y-coordinate.
- "IDMMX.DM_getQualities(regModel): This command is the API equivalent of the Qualities Extractor operator. It returns a 2-column table (QUALITYNAME and QUALITYVALUE) in which each row contains one of the quality numbers model quality Q, accuracy A, ranking quality R, or reliability Rel.
- "IDMMX.DM_getFields(clasModel): Returns a 4-column table (COLNAME, FIELDNAME, MININGTYPE, and IMPORTANCE) that contains for each active field in the model the name of the data column to which it refers, its alias name, its mining type (0 for categorical or 1 for numeric), and its importance, a value between 0 and 1. The sum of all importance values is 1.
- "IDMMX.DM_getCorrelations(clasModel): Returns a 3-column table (FIELDNAME1, FIELDNAME2, and CORRELATION) that lists all field-field correlation numbers contained in the model. Correlations between continuous numeric fields are linear correlation coefficients, correlations between discrete fields are adjusted contingency coefficients, and correlations between discrete and continuous fields are not calculated.

Applications of Transform Regression

Transform Regression is a powerful general purpose numeric prediction algorithm. It produces good models on almost any kind of input data. The method has no algorithm-specific tuning parameters. Typical application scenarios are:

- Target marketing: Select the most profitable customer group for a marketing campaign.
- Predict sales.
- Predict revenue, cost. or profitability.
- Risk assessment: Predict credit default risk and predict insurance claim risk.
- Fraud detection: Find those transactions that are most likely fraudulent.
- Quality assurance: Predict the risk of failure.

Compared to other prediction algorithms, such as Decision Trees or Linear Regression, Transform Regression has the disadvantage of not producing an easily interpretable model from which easy rules can be derived. Therefore, Transform Regression should be used whenever a numeric value has to be predicted. If you want to know why a value was predicted, or if you want to learn more about the hidden rules in your data that influence the target fields, you should better use Linear or Logistic Regression, or you should discretize the target field and use Tree Classification.

Further reading

- Apte, et al, "A Probabilistic Estimation Framework for Predictive Modeling", IBM Systems Journal, 41, No. 8, 2002.
- Pednault, "Transform Regression and the Kolmogorov Superposition Theorem", Proceedings of the Sixth SIAM International Conference on Data Mining, 2006.
- Dorneich, et al, "Embedded Predictive Modeling in a Parallel Relational Database", Proceedings of the 21st ACM Symposium on Applied Computing, 2006.

All papers listed above are available online from:

http://domino.research.ibm.com/comm/research_projects.nsf/pages/dar.pubs.html

Scoring a regression model

Once a regression model has been created and visually checked, you might want to apply it to new data for which the target field value is not yet known. Applying a regression model to a table means predicting the target field value for each record. In addition to the predicted value, you can get other scoring results, for example, the predicted incertitude range of the prediction.

For this purpose, the Design Studio provides the Scorer operator. This operator takes two inputs, the model and the table to which the model is to be applied. In Figure A-34, we have connected a Scorer operator to a regression model that predicts the number of claims that an insurance customer is likely to generate within a certain period of time. The model calculates the prediction based on patients' demographic data, such as age or sex, and on insurance history data. The table matches with the model and provides this required information.

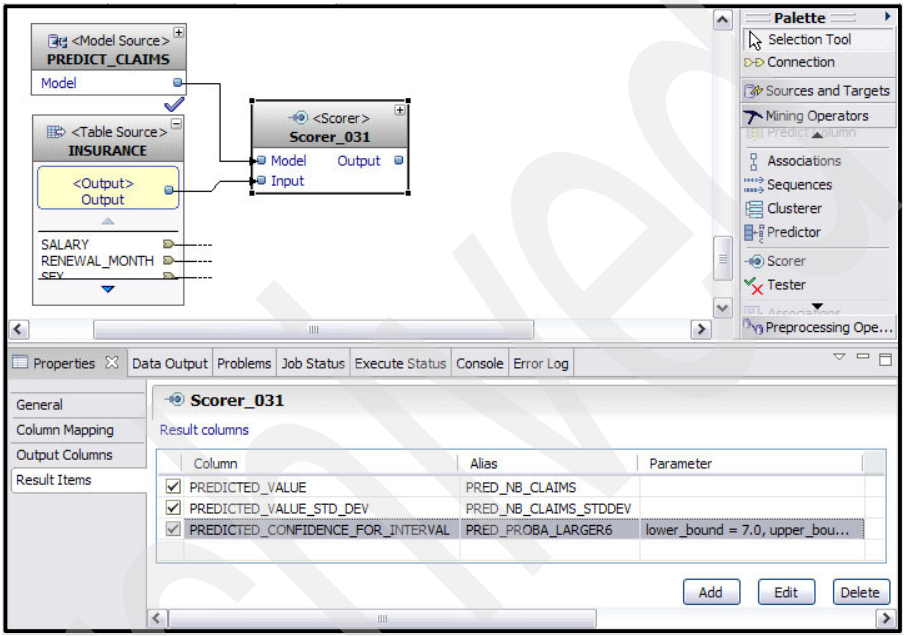


Figure A-34 Scoring a r egression model

In the Properties view of the Scorer operator, you can select which scoring results will be generated. Per default, for each data record the predicted value and the predicted incertitude of the prediction are activated. You will find these columns marked in the Result Items window of the Scorer Properties view. Using the Add button, you can activate a third result measure, the predicted probability that the target value lies in a certain interval. We have chosen the interval [7, 1000]. Furthermore, we have redefined the names of the data columns that correspond to the three measures. Per default, they are labeled PREDICTED_VALUE, PREDICTED_VALUE_STD_DEV, and PREDICTED_CONFIDENCE_FOR_INTERVAL. We have changed that to the names PRED_NB_CLAIMS, PRED_NB_CLAIMS_STDDEV, and PRED_PROBA_LARGER6.

Now we have to provide a table into which the scoring results will be written. By default, the scorer operator tries to write all columns of the input table plus the activated scoring result measures into the result table. If you do not want to reproduce the input table content in the result, you can deactivate certain output columns in the Output Columns window of the Scorer's Properties view.

In many cases, a result table of exactly matching column format does not yet exist. Instead of manually creating a new table, you can right-click the output port of the Scorer operator and select the menu item **Create suitable table**. A wizard pops up in which you are presented with a set of default settings for the new table. You can modify these default settings. In Figure A-35, we have marked the column RECORDID of the new table as primary key. All other settings are the default settings.

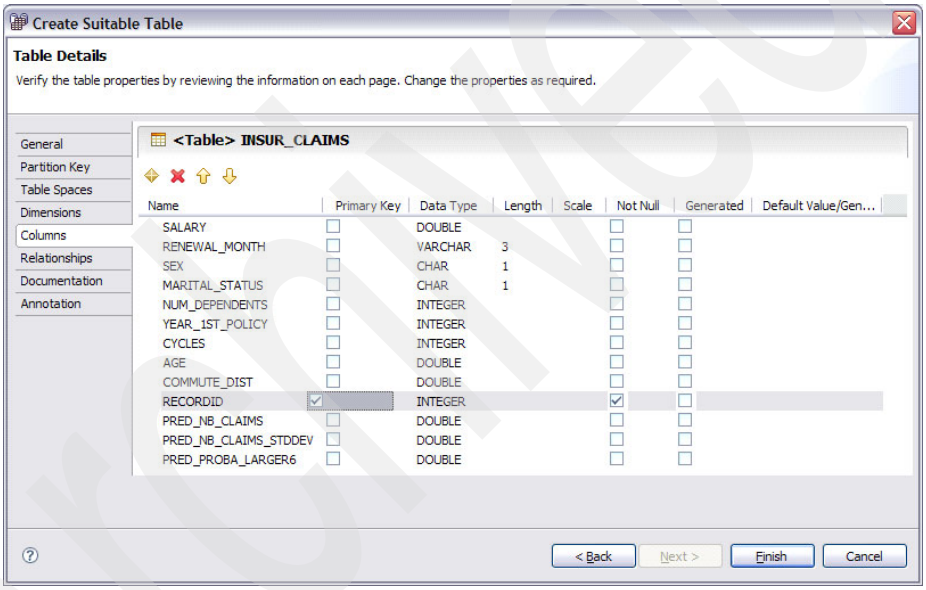


Figure A-35 Table for Scoring results

Now we can click the **Finish** button of the table creation wizard, and the resulting mining flow, as depicted in Figure A-36, has the desired form. We can execute this flow and fill the table INSUR_CLAIMS with the desired scoring results.

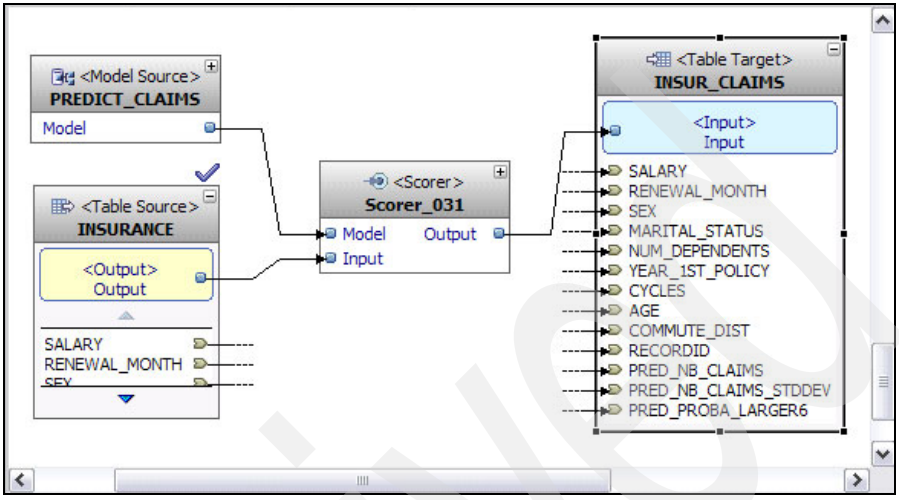


Figure A-36 Mining flow

Basic scoring result measures

The graphical panels of the Scorer operator allow modifying and activating the most commonly used result measures of regression scoring.

Predicted Value

This is the predicted target field value for the current data record that is calculated based on the record's existing field values.

Predicted Standard Deviation

This measure is the predicted incertitude of the predicted value. The Transform Regression kernels reserves a part of the training data for model validation. These data are not used for creating the model but for comparing its predictions to the actual target values. These comparisons provide several statistical distributions of prediction errors, depending on the value ranges of certain input fields. The Predicted Standard Deviation is the standard deviation of the statistical error distribution for the input field value range into which the current data record falls.

For the other DB2 Warehouse regression kernels, Linear and Polynomial Regression, this scoring result measure is not available.

Predicted probability

To predict the probability that the target value lies in a given value range, let [lower,upper] be the user-specified value range. Then the probability measure is calculated as the area under the Gaussian curve with mean value equal to 'Predicted Value' and standard deviation equal to 'Predicted Standard Deviation' between the values lower and upper. Hence, when calculating this measure, we implicitly assume that the prediction errors are normally distributed.

Advanced scoring result measures

Using Custom SQL and the DB2 Warehouse SQL Mining API, you can access more advanced scoring results in addition to the ones described above. The Mining API defines a DB2 user defined type (UDT) called DM_RegResultSpec that provides a method for retrieving the predicted range in which the target value lies with an arbitrary user-specified confidence:

```
DM_getPredDev(confidence)
```

Returns the half width w of the value range around the predicted value p in which the true target value t lies with a confidence level of confidence:

```
conf( |p - t| ≤ w )  = confidence
```

This measure is calculated based on the assumption that the model's prediction error is normally distributed and unbiased.

The basic result measures described in the preceding section are accessible in the API through:

- `DM_getPredValue`
- `DM_getPredStdDev`
- `DM_getConfidence`

In Example A-3, we show an SQL code that could be embedded into a Design Studio workflow by means of the Custom SQL operator. The code calculates for each data record, which represents an insurance customer, the confidence (probability) that the customer will generate more than seven insurance claims within a certain time period.

Example: A-3 Custom SQL code example

```
WITH "RESULTSPECIFICATION"( "RESULTSPEC" ) AS
 ( VALUES( IDMMX.DM_RegResultspec() )),
  "REGRVIEW"( "CUSTOMER_ID", "REGRRESULT" ) AS (
    SELECT "CUSTOMER_ID",
    IDMMX.DM_applyRegModel (
      'PredictClaims',
      rec2xml( 1.0, 'COLATTVAL', '',
       B."AGE",B."GENDER",B."MARITAL_STATUS",B."PROFESSION",
       B."INCOME",B."ZIP_CODE" ),
      R."RESULTSPEC"
    )
    FROM "CUSTOMERS" B, "RESULTSPECIFICATION" R
  )
 SELECT
  "CUSTOMER_ID",
  CAST( IDMMX.DM_getConfidence("REGRRESULT",7,1000)
    AS DEC(5,4)) AS "PROBA_NBCLAIMS_LARGER6"
 FROM "REGRVIEW";
```

The two commands shown are commands of the Mining API. The first command, DM_applyRegModel, reads one data record and calculates its scoring results using the regression model with name *PredictClaims*. The content of the data record is provided through the **rec2xml** statement. The second API command, DM_getConfidence(…,7,1000), retrieves the confidence that the current customer will produce seven or more claims.

Associations

The Associations mining function finds items in your data that are associated with each other in a meaningful way.

Introduction and notations

An item can be an arbitrary “atomic” event, article, fact, or component in the data. A prerequisite for finding associations between these atomic items is that a grouping and assignment of several of the items to one comprising and superior entity exists. An item set is such a group of items that belong to the grouping entity. Often, this grouping entity is called a transaction (TA). An association or association rule is a combination of two item sets in the form of a rule 'itemset1 ==> itemset2'. The left hand side of the rule is called the rule body or antecedent, the right hand side the rule head or consequent.

Table A-4 lists typical use cases of mining for associations.

Table A-4 Typical uses for mining associations

Industry	Use case	Grouping entity	Typical body items	Typical head items
Retail	Market basket analysis	Bill ID or purchase ID	A purchased article	Another purchased article
Mfg	Quality assurance	Product (for example, vehicle ID)	Component, production condition	Problem, error ID
Medicine	Medical study evaluation	Patient or test person	Single treatment info	Medical impact

Typical association rules for the three use cases listed in the table are:

- {milk $\wedge$ baby food} ==> {diapers }
- {axles=17-B $\wedge$ engine=AX-Turbo 2.3 $\wedge$ production_chain=3} ==> {delay > 2 hours}
- {Hypomed Forte 5 ml $\wedge$ Placebon Extra 2ml} ==> {patient cured}

Associations rule properties

An Associations rule can be characterized by the following properties:

- The contained items (in the rule body, in the rule head, and in the entire rule).
- Categories of the contained items. Often, an additional hierarchy or taxonomy for the items is known. For example, the items 'milk' and 'baby food' might belong to the category 'food', 'diapers' might belong to the category 'non-food'. 'axles=17-B' and 'engine=AX-Turbo 2.3' might be members of the category 'component', 'production_chain=3' of category 'production condition', and 'delay > 2 hours' of category 'error state'. Hence, the second sample rule can be characterized by the fact that its body contains components or production conditions, and its head an error state.
- The support of the rule. Absolute support is defined as the total number of group entities (transactions) for which the rule holds. Support or relative support is the fraction of all transaction for which the rule holds.
- The confidence of the rule. Confidence is defined as:

  confidence := support(body==>head) / support(body)
- The lift of the rule. Lift is defined as:

  lift := support(body==>head) / (support(body) • support(head))

  lift>1 means that the rule 'body==>head' appears more frequently than expected assuming that 'body' and 'head' are statistically independent.
- The weight (cost,price) of the rule. Often, an economical value measure exists for each item, for example, its price or cost. If this is the case, the aggregated weight (cost or price) of an association rule can be defined as the sum of all item weights multiplied with their multiplicities. For example, if those transactions that support the association rule '{milk $\wedge$ baby food} ==> {diapers }' contain on average 2.3 cartons of milk at $1 each and 4.7 portions of baby food at $0.80 each and 1.2 packets of diapers at $9.90, the average weight of the association rule is $15.56.
- The weight (cost,price) of the transactions that support the rule (TA weight). If an economical value measure exists for each item, a second weight property for a rule in addition to its average weight can be given: the average weight of the transactions that support the rule. If, for example, in those purchases that contain the items 'milk', 'baby food' and 'diapers', and also contain other items at an average price of $34.13, the TA weight of the rule '{milk $\wedge$ baby food} ==> {diapers }' is $49.69.

Data formats

In this section, we discuss the data formats for Associations mining.

The transactional or pivoted data format

Often, the input data for Associations mining are available in a two-column format in which one column is the group field and contains transaction IDs, and the second field is the item field and contains items. For example, an input table for market basket analysis could be as depicted in Table A-5.

Table A-5 Input table example

Trans ID	Item
0000001	Diet Coke
0000001	Toast
0000002	Milk
0000002	Cheddar
0000002	Beer
0000003	Dark Bread

Here, the first purchase contains the articles 'diet coke' and 'toast', the second one the items 'milk', 'Cheddar', and 'beer'. For this data format, you have to specify the group column, in this example TRANSID, in the Properties view of the Associations operator, as depicted in Figure A-37.

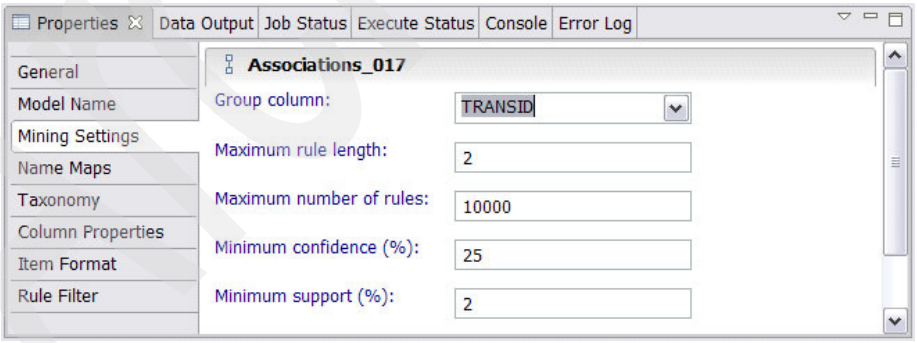


Figure A-37 Associations operator: properties view

The unpivoted or broad data format

In other use cases, the data is available in a format in which each row in the table corresponds to one transaction. For example, a bank could have the following data on the customers and the banking services that are using the data, as depicted in Table A-6.

Table A-6 Banking services example

Cust ID	CreditCard	Account	Avg.Balance	Loan	RiskClass
0017854	Gold	Online	12539.78	Yes	4
0017855	-	Standard	396.15	No	3

In this case, no group field is needed because no transaction spans more than one table row. In the Properties view of the Associations operator, the default content of the field 'Group column', <none>, can remain unchanged. However, for the data table shown above, the key column CUST_ID is not a suitable item column. Therefore, its usage type should be set to inactive, as depicted in Figure A-38.

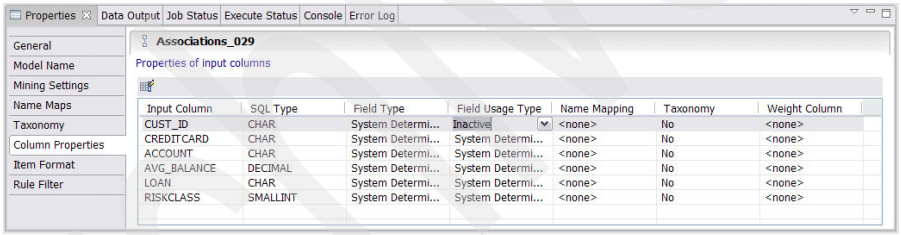


Figure A-38 Properties view of associations operator

More generally, by default all input table columns that are not flagged as group columns are interpreted as item columns. Therefore, each column whose content should not appear in the generated associations has to be switched to 'inactive'.

Item formats

Independently of the global data table format (transactional with group field or broad without group field), each single item column can have differently formatted content. Furthermore, for each item column, the way the contained items should appear in the generated associations can be modified. You can adjust both aspects of the item formats to your data and your needs. There is a coarse-grained way of adjusting item formats by means of a graphical GUI window in the Associations operator Properties view, and a fine-grained way using textual commands in the Optional Parameters text field.

The Properties window Item Format

The window Item Format of the Associations operator Properties view allows you to specify the way items names should appear in the detected Associations, as depicted in Figure A-39.

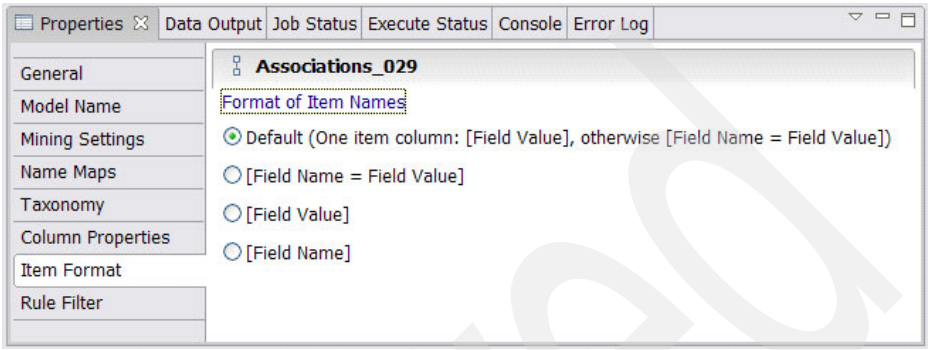


Figure A-39 Properties window: Item Format

In the associations, an item can appear as the content of an item column in the input data (Field Value), as the name of the item column (Field Name) or as a combination (Field Name = Field Value).

If your data contains only one item column, the name of the column is in most cases not interesting because it is the same for all items. Therefore, the default item format in this case is 'field value'. From our two-column sample table containing supermarket purchases, the following rule could be generated in this format:

```
Cheddar ==> beer
```

If we set the item format to 'Field Name = Field Value', this would become:

```
ITEM=Cheddar ==> ITEM=beer
```

And if we try to activate the format 'Field Name', we get an error message when we try to start the mining flow because this format would create the useless rule format:

```
ITEM ==> ITEM
```

For our second sample table, the default format is 'Field Name = Field Value', creating rules such as:

```
CREDITCARD=gold ==> ACCOUNT=online
```

For this table, no other item format makes sense. But now imagine we have a slightly different table, as depicted in Table A-7.

Table A-7 Banking services example

Cust_ID	CreditCard	Online_ Account	High _Balance	Loan	Low_Risk
0017854	Yes	Yes	Yes	Yes	No
0017855	No	Yes	No	No	Yes

In this table, all columns are binary, and the main item information is contained in the column names. Therefore, for this table, the format 'Field Name' is most appropriate. This format produces rules such as:

```
CREDITCARD ==> ONLINE_ACCOUNT
```

which is a short form for:

```
CREDITCARD=yes ==> ONLINE_ACCOUNT=yes
```

Besides creating more readable rules, the format 'Field Name' has another beneficial effect: negative rules, such as:

```
CREDITCARD=no ==> ONLINE_ACCOUNT=no
```

are automatically suppressed. More generally, in the item format 'Field Name', the following item values are interpreted as 'is not present': SQL NULL, 0, '-', ' ' (blank), '' (empty string), every string starting with F or f (such as 'false') or N or n (such as 'no'). All other values are interpreted as 'is present'. Only entries that are present contribute to association rules.

The command DM_setItemFormat

The command DM_setItemFormat, typed into the Optional Parameters field, provides DB2 Warehouse's full range of item format adjustment functionality. Using this command, you can:

- Set different item formats for different item fields.
- Activate a fourth format, the so-called set-valued item format.
- Toggle which item entries are interpreted as 'is not present'.
- The window Item Format of the Association operator's Properties view allows specifying. For example, there are two versions of the command:

  ```
  DM_setItemFormat(formatID)
  ```

  or

  ```
  DM_setItemFormat('colName',formatID)
  ```

The first version without a colName argument redefines the item format for all item fields, the second version only for the item field 'colName'. Both versions can be concatenated, but the version without field arguments should be placed first, because it overwrites all previous format settings. The following formatID values are recognized:

- 0: Format 'Field Name = Field Value'.

  Item field values equal to SQL NULL, '-', or '' (empty string) are neglected.

- 1: Format 'Field Value'.

  Item field values equal to SQL NULL, '-', or '' (empty string) are neglected.

- 2: Format 'Field Name'.

  Item field values equal to SQL NULL, 0, '-', ' ' (blank), '' (empty string), or every string starting with F or f (such as 'false') or N or n (such as 'no') are neglected. All other item field values are interpreted as 'is present'.

- 10: Format 'Field Name = Field Value'.

  Item field values equal to SQL NULL, '-', or '' (empty string) are not neglected, which means they can appear in the generated association rules.

- 101: Multi-value format. This format expects an XML input string of the form:

  ```
  <item>x1</item><item>x2</item>...<item>xn</item>,
  ```

  where x_1, x_2, ... , x_n are arbitrary different item values. This format interprets the XML string as set of separate item values. The created associations are of the format Field Value. For example, the input '<item>milk</item> <item>Cheddar</item> <item>beer</item>' contributes to the rule milk ==> beer.

  The input format containing several <item>…</item> tags has one slight disadvantage: a single root XML element is missing, therefore the strings have no well-formed XML documents. If the input is generated by an XML tool, that can be a problem. Therefore, DB2 Warehouse also supports the following two variants for multi-valued input (where xxx stands for an arbitrary encoding):

  ```
  '<itemset><item>milk</item><item>Cheddar</item>
  <item>beer</item></itemset>'  and
  '<?xml version="1.0" encoding="xxx" ?>
  <itemset><item>milk</item><item>Cheddar</item>
  <item>beer</item></itemset>'
  ```

Note: In DB2 Warehouse versions higher than 9.1.2, an item field type can be set to 'multi_categorical' in the Column Properties view. If this is done, the field's default item format automatically becomes 101 (Multi-value).

Figure A-40 shows how the default item format for the first sample table on bank customers can be modified using DM_setItemFormat.

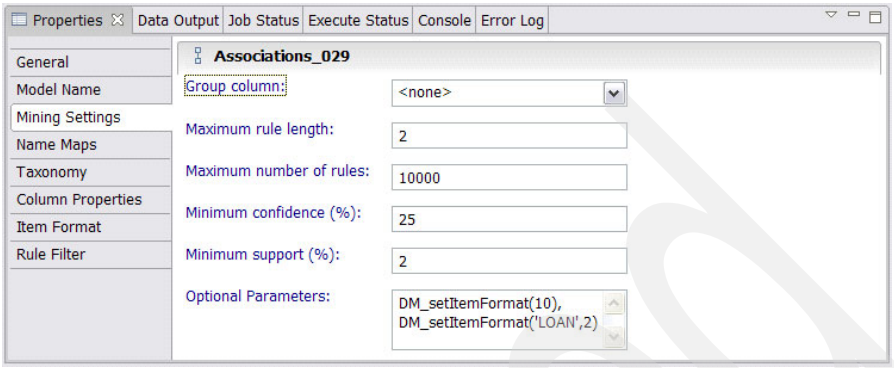


Figure A-40 Modifying values

For Table A-8, the default item format is 0. The optional parameters below change the format of the column LOAN, which is binary, to 'Field Name'. The format of all other item columns is changed to the nonstandard 'Field Name = Field Value' variant, which does not neglect SQL NULL, '-' or empty strings.

Table A-8 Banking customer sample table

Cust_ID	CreditCard	Account	AvgBalance	Loan	RiskClass
0017854	Gold	Online	12539.78	Yes	4
0017855	-	Standard	396.15	No	3

The multi-valued item format and text analysis

The item format 101 is particularly helpful when mining for associations is combined with text analysis. Assume that we have a table with structured medical patient information such as age, gender, cholesterol, blood pressure, heart rate, and other structured medical data. In addition, for each patient we have a free-text patient history, for example:

> Mr. S. is a heavy smoker, his physical fitness is very low. He says he is very short of breath after climbing a few stair steps. He frequently suffers from headaches. He has a stressful job in the middle management of an IT company.

Using text analysis tools, we could extract keywords or concepts from this text that could have an impact on the medical state of the patient, for example, concepts such as 'smoker', 'headache', or 'shortness of breath'. When analyzing a given free-text patient record, we do not know beforehand how many keywords are detected in this text. It could be 1, 17, 352, or none at all. Therefore, the multi-valued item format is very well suited in this case, because the result of the text analysis can be written into one single item field, independently of the actual number of detected keywords. From our example text, the text analysis tool could create the multi-valued column entry:

```
<item>smoker</item><item>shortness of breath</item>
<item><headache></item>
```

The file ../dwe/im/IMinerX/V9.x/samples/ModelingDB2/heart.data contains a medical data set as described above. You can load that data set into the table HEART by issuing **db2 -tf heartImport.db2** from the DB2 command window. In the table HEART, the column MEDICAL_HISTORY contains the original free text, and the column KEYWORDS contains the extracted keywords. Figure A-41 shows how the multi-valued character of the item field KEYWORDS can be communicated to the Associations operator using the Optional parameters field.

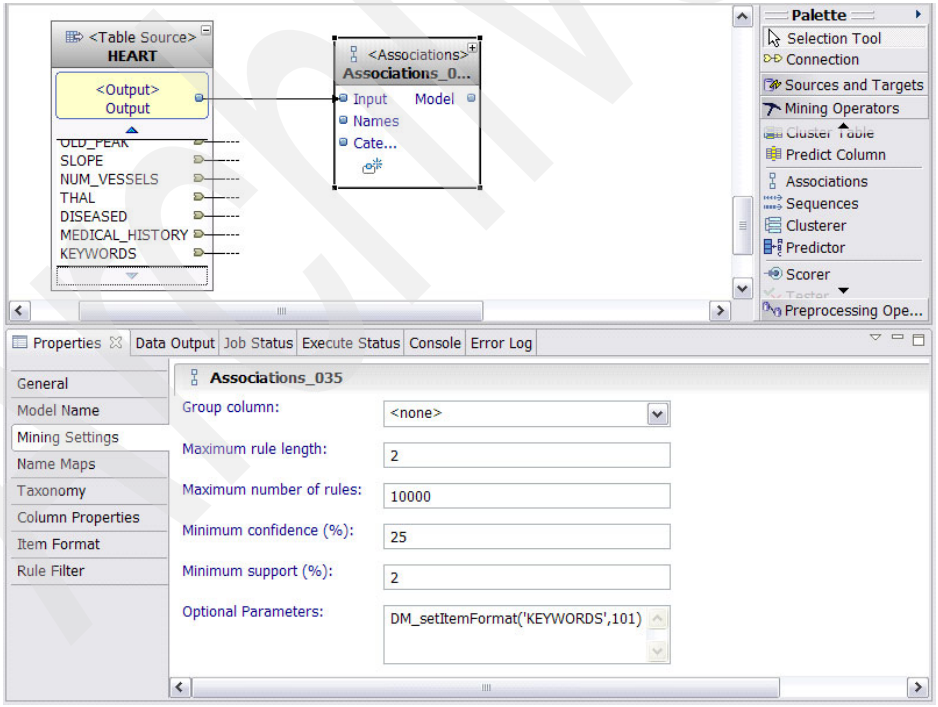


Figure A-41 Using the optional parameters text box

Name mappings and taxonomies

Often the value of the associations mining result can be increased if, together with the main transactions data source, two auxiliary data sources are used.

Name mappings define cleartext names for the items in the main transaction table if the latter in their original form are unintuitive numeric IDs. This helps to create more understandable association rules.

For example, the market basket analysis rule '00317-0025 ==> 00317-0109' is not very helpful. But knowing that the cleartext name of item 00317-0025 is 'low-fat milk, 1l' and 00317-0109 is 'cream 30%, 250ml', the same rule becomes much clearer: 'low-fat milk, 1l' ==> 'cream 30%, 250ml'.

A name mapping table is a two-column table that contains an “original item” column and a “cleartext description” column. In the DesignStudio, the name mapping table can be defined as Table Source operator and connected to the Associations operator's input port called Names. If more than one name mapping is to be used, additional input ports for the Associations operator can be created by clicking the symbol at the bottom of the Associations operator box, as shown in Figure A-41 on page 446. Once all name mapping tables are connected to the Associations operator, they must be configured in the Name Maps window of the operator's Properties view. That means that you must specify which column contains the original item name and which one contains the cleartext name. This is depicted in Figure A-42.

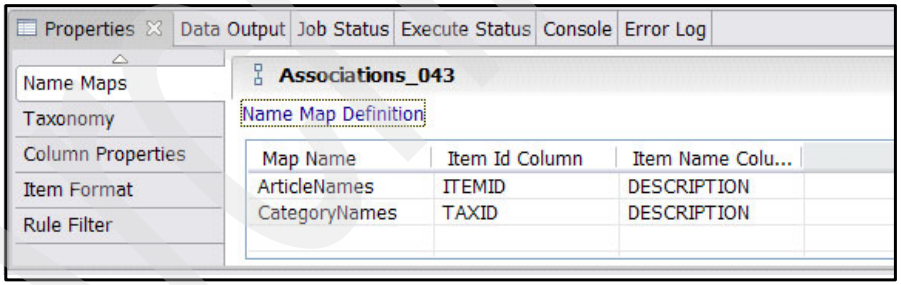


Figure A-42 Name maps: Property view

Taxonomies define hierarchies or categories for the items that are contained in the transaction data. This information can be used for powerful filter criteria and for creating more general sequences that cover not single items but entire item categories.

A taxonomy table is a two-column table that contains a parent and a child column. The child column contains the more concrete part of the relationship, the parent column the more generic part. For example, ('milk 1.5% fat', 'milk products') could be one child-parent pair and ('milk products', 'food') another one.

In the DesignStudio, the taxonomy table can be defined as Table Source operator and connected to the Associations operator's input port called Categories. If more than one taxonomy table is to be used, additional input ports for the Associations operator can be created by clicking the appropriate symbol at the bottom of the Associations operator box, as shown in Figure A-41 on page 446. Once all taxonomy tables are connected to the Associations operator, they must be configured in the Taxonomy window of the operator's Properties view. That means that you must specify which column is the parent column and which one is the child column. This is depicted in Figure A-43.

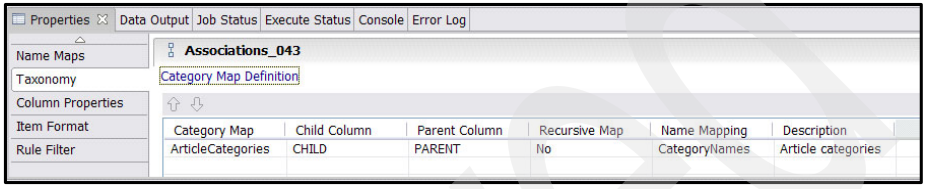


Figure A-43 Operator Properties view: Taxonomy

When all name mappings and taxonomies have been configured, you have to activate them for one or more item columns. This is done in the Column Properties window of the Association operator's Properties view. Figure A-44 on page 449 shows an associations mining flow that uses a transaction table RETAIL, a taxonomy table RETAIL_TAX, a name mapping table RETAIL_NAMES for the original items, and a second name mapping table RETAIL_TAXNAMES, which contains cleartext names for the item categories from the taxonomy table.

Note that the first name mapping and the taxonomy are activated for the item column ITEMID. The second name mapping has been activated for the taxonomy inside the Taxonomy window.

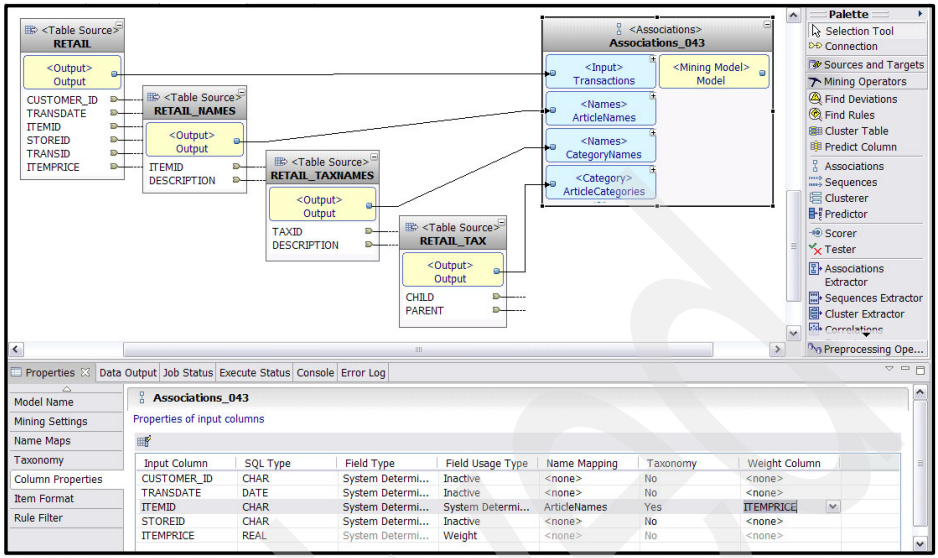


Figure A-44 Column properties

Specifying weights: cost or price

In most cases, you have a certain economic motivation for running a mining for associations. It is therefore desirable that the detected associations can directly express the economic value that they represent. With associations, you can do this if you specify so-called weight information for the involved items. The weight of an item can be any numeric measure representing the importance of an item, such as its price, cost, physical weight, size, area, and duration. In business applications, cost or price are the most frequently used weights.

Weight information can be available in two different ways:

1. Each item has a fixed weight. Then a separate weight table can be given analogously to a name mapping table. The table assigns one fixed weight value to each item.
2. The weight of the items is not fixed but can change from one transaction to the other. An example of this situation is the selling price of articles in a supermarket. These prices change over time, for example, due to special sales campaigns. In this case, the weight information has to be given in a separate data column in the main transaction table, and not in a separate weight table.

Specifying a weight column in the main transactions table

Figure A-44 on page 449 shows such a case. The main transactions table RETAIL contains not only the item column ITEMID but also a parallel weight column ITEMPRICE. In Figure A-44 on page 449, you can see how this weight column is activated for mining in the Column Properties window of the Associations operator: The field usage type of column ITEMPRICE is set to 'Weight', and then this weight column is activated for the item field ITEMID.

Specifying a separate weight table

Separate weight tables are defined and associated with a certain item field by typing the following commands into the Optional parameters window of the Associations operator:

```
DM_addWeight('weightName','tableName','itemCol','weightCol')
DM_setFldWeight('fieldName','weightName')
```

Figure A-45 shows how this appears if the item field in the transaction table is ITEMID and if the weight info resides in the table PRICES that contains the columns ITEM and PRICE.

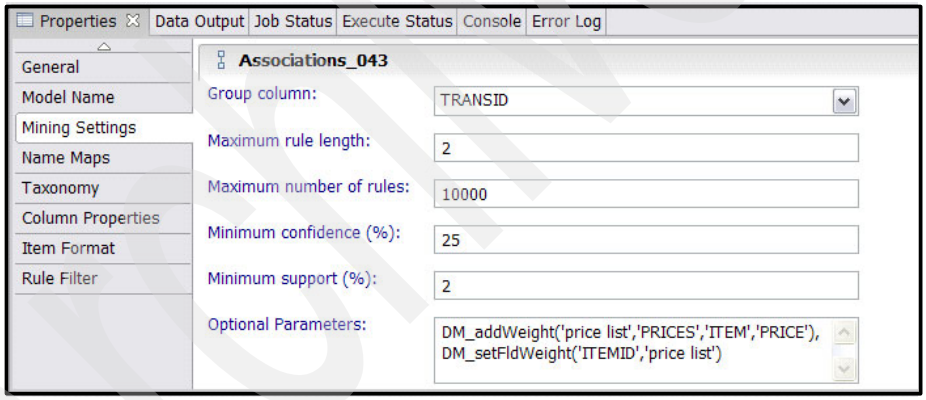


Figure A-45 Optional parameters

How weight info is used during the mining

Once you have specified the weight info for the associations mining, this info will be used in two ways.

First, you can define several weight-related rule filters. These filters can be applied during model creation, in order to detect only those association rules that fulfill certain weight criteria. The filters can also be applied as “post-filters” on an existing model in order to produce a filtered submodel. Both aspects of filtering are described in more detail in “Rule filters” on page 454.

Second, the generated Associations model contains weight statistics, consisting of mean value, standard deviation, minimum value and maximum value, for two different kinds of rule-related weights:

1. The weight (cost,price) of the association rule.

   The aggregated weight of an association rule can be defined as the sum of all item weights multiplied by their multiplicities. For example, if those transactions that support the association rule '{milk $\wedge$ baby food} ==> {diapers }' contain on average 2.3 cartons of milk at $1 each and 4.7 portions of baby food at $0.80 each and 1.2 packets of diapers at $9.90, the average weight of the association rule is $15.56.

2. The weight (cost,price) of the transactions supporting the rule (TA weight).

   This is the average weight of the transactions that support the rule. If, for example, in those purchases that contain the items 'milk', 'baby food' and 'diapers', there are also other items at an average total price of $34.13, the TA weight of the rule '{milk $\wedge$ baby food} ==> {diapers }' is $15.56 + $34.13 = $49.69.

In the Associations visualizer, only the mean weight of the rule is displayed by default. You can add arbitrary combinations of the remaining seven weight statistics measures (standard deviation, minimum and maximum of the rule weight, plus the four TA weight measures) by editing either the Visualizer Properties $\rightarrow$ Rules Columns window or the Model Properties $\rightarrow$ Rules Columns window. In the first case, the modified settings will be applied to all associations models, and in the second variant only to the current one.

In Figure A-46, we have added the mean value of the TA weight column to the displayed result.

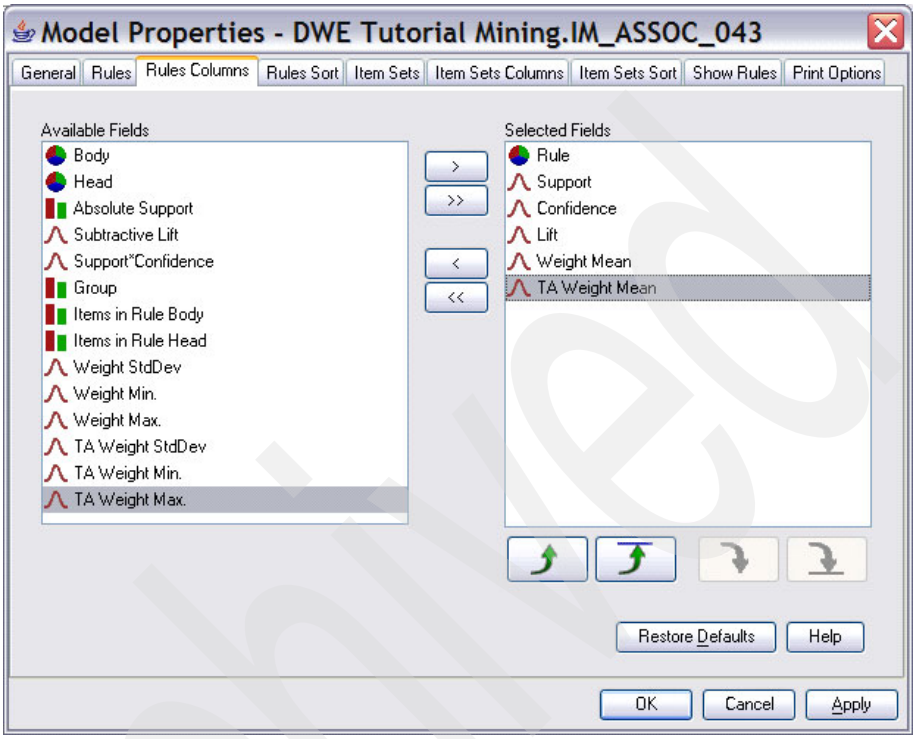


Figure A-46 Associations visualizer

Specifying the discretization mode for numeric fields

Whenever the input data for associations mining contains numeric fields, in particular floating point fields, this data must be discretized, because otherwise each single numeric value would appear only once or a small number of times in the data, and no such specific value would reach the required minimum support threshold. Associations automatically perform this discretization, called *binning*. You can modify the default binning method and adjust it to your needs. To that purpose, you can use the commands DM_setAlgorithm, DM_setFldNumBins, DM_setFldOutlLim, and DM_setPowerOptions in the Optional parameters text field of the Association operator's Properties view.

Default binning mode: (μ,σ)-based binning into five bins

By default, each numeric item field, which means each item field of floating point or integer type, which contains more than 100 different values, is automatically discretized into five bins. The bin boundaries are determined as follows:

- Up to 10000 training data records are read, providing 10000 numeric values of the item field to be binned.
- From these 10000 values, mean (μ) and standard deviations (σ) are calculated.
- The raw (μ,σ) values are slightly rounded to avoid many trailing digits. For example, (μ,σ)=(71.64198, 1.36812) might be smoothed to (μ,σ)=(71.6, 1.4)
- Based on the smoothed (μ,σ) values, the following bins are created:
  -] $-\infty,\mu$ $-3\sigma/2$] (=strongly below average)
  - [$\mu-3\sigma/2$, $\mu-\sigma/2$ [(= value below average)
  - [$\mu-s\sigma/2$, $\mu+\sigma/2$ [(= value close to average)
  - [$\mu+\sigma/2$, $\mu+3\sigma/2$ [(= value above average)
  - [$\mu+3\sigma/2$, $+\infty$ [(= value strongly above average)

User-defined modifications to the (μ,σ)-based binning mode

- The default number of records to be read for the determination of (μ,σ) can be changed using the parameter <NumSampleRecordsForBins> using the method DM_setAlgorithm.

  For example, the following command increases the number of records to be read for calculating (μ,σ) to 20000:

  ```
  DM_setAlgorithm('SIDE','<NumSampleRecordsForBins>20000
    </NumSampleRecordsForBins>')
  ```

- The default number of bins (which is five) can be changed for all numeric fields at once using the parameter <NumBins> in the method DM_setAlgorithm.

  For example, the following command increases the default number of bins to 12, and it increases the number of records to be read for (μ,σ) to 20000:

  ```
  DM_setAlgorithm('SIDE','<NumBins>12</NumBins><NumSampleRecordsForBin
  s>20000</NumSampleRecordsForBins>')
  ```

- If you want to change the number of bins for single item fields independently of other item fields, you can use the method DM_setFldNumBins. For example, the following command increases the number of bins for the field INCOME to 20:

  ```
  DM_setFldNumBins('INCOME',20)
  ```

Note: Depending on the actual number N of bins specified for a given item field, the bin widths and boundaries are adjusted as follows:

- If N < 8: bin width = σ, bins are symmetrically centered around μ.
- If 7 < N < 15: bin width = $\sigma/2$, bins are symmetrically centered around μ.
- If N > 14: bin width = $\sigma/4$, bins are symmetrically centered around μ.

User defined mode with equidistant bins

- If you want to have equidistant bins between user-specified lower and upper boundary values, you can use the command DM_setFldOutlLim. If you specify an upper and a lower limit, the two outmost bins will be created below the lower and above the upper limit. The remaining bins will be formed equidistantly between the limits. For example, the following command defines N-2=3 equidistant bins with boundaries 10000, 30000, 50000, 70000 plus two outlier bins (<10000; >70000) for the field INCOME. (Remember that N=5 is the default setting):

```
DM_setFldOutlLim('INCOME',10000,70000)
```

- You can combine the commands DM_setFldOutlLim and DM_setFldNumBins to modify the number of equidistant bins. For example, the following commands create six equidistant bins of width 10000 between 10000 and 70000 plus two outlier bins for the field INCOME:

```
DM_setFldOutlLim('INCOME',10000,70000)
DM_setFldNumBins('INCOME',8)
```

Modifying when the binning starts

By default, the binning starts when a numeric field has more than 100 different values. This threshold of 100 can be modified by using the power option parameter -MAX_DISCR_COUNT. For example, the following command triggers the onset of binning when more than 20 different values have been found:

```
DM_setPowerOptions('-MAX_DISCR_COUNT 20')
```

Rule filters

A frequent challenge in mining for associations is that too many rules are found, and it is difficult to detect the really interesting ones within thousands or tens of thousands of other rules that are either trivial or not helpful for the specific use case. Therefore, it is often necessary to define a set of rule filter criteria before starting the mining run. These criteria help both to keep the resulting set of associations manageable and to speed up the training process.

Associations mining supports very flexible and powerful rule filters. All rule properties listed in “Introduction and notations” on page 438 on Associations mining can be used as filter criteria, and all filter criteria can be combined by arbitrary logical AND and OR operators.

Furthermore, DB2 Warehouse supports the application of rule filters both during model creation (pre-filtering) and to extract submodels out of existing models (post-filtering).

Rule filters as pre-filters

The Associations operator Properties view offers two windows for defining rule filters. First, the Mining Settings window, depicted in Figure A-47, offers the ability to define some very common filter criteria that should be used in almost any associations mining use case. They have a significant impact on keeping the run times of the Associations operator in an acceptable range.

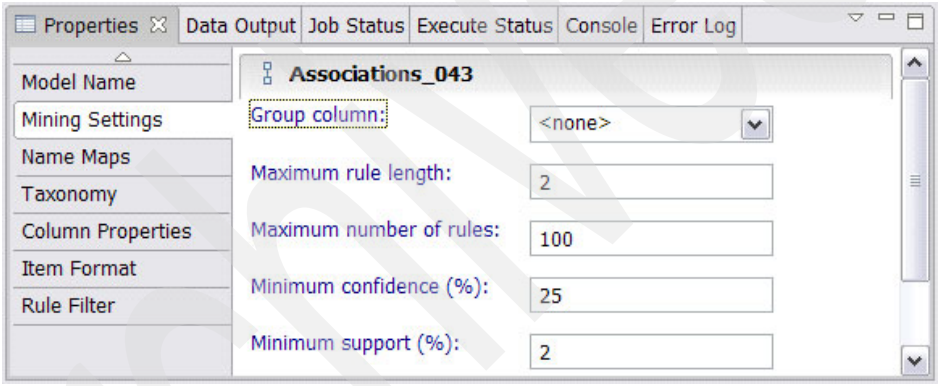


Figure A-47 Mining settings

By default, the following filters are applied:

- The maximum rule length is 2, which means you will only get associations with one item in the rule body and one item in the rule head.
- The maximum total number of rules is 10000. If more than 10000 rules are found, the 10000 with highest support are stored in the generated associations model. In the Associations operator depicted above, we have reduced this default value to 100.
- The minimum rule confidence is 25%.
- The minimum rule support is 25%.

Minimum support is the filter criterion that most drastically affects the runtime of the model training process if it is not specified with care. Therefore, if you delete the default value of 2% and leave the Minimum support field empty, the Associations operator still applies a fallback minimum support criterion that corresponds to an absolute support of five occurrences. The idea behind this fallback setting is that if a rule that is supported by less than five transactions cannot be statistically significant, it is probably purely random noise.

There are certain use cases in which even this fallback support is too high, for example, if you are looking for rare deviations. In this case, you have to type in a very small percentage, for example 0.00001, into the Minimum support field to overcome the barrier of minimum absolute support = 5.

For defining more complex filter criteria, the Properties view of the Associations operator offers a separate window called Rule Filters.

If you open this window, you see a listing of all non-default rule filters defined so far. In our example, depicted in Figure A-48, only the Maximum number of rules has been changed.

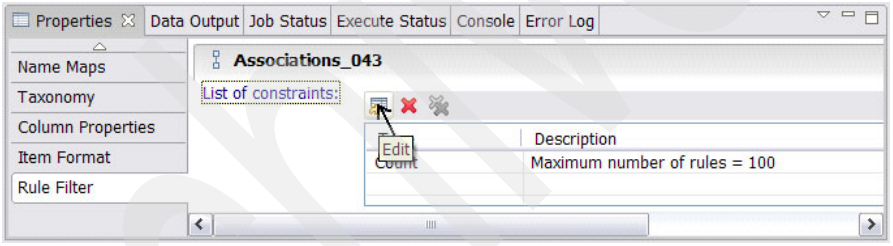


Figure A-48 Rule Filters

If you click the **Edit** symbol in this listing, a new window with three tabs containing detailed range-based, count-based, and item-based filters pops up. In Figure A-49, we have entered some more sophisticated filter criteria.

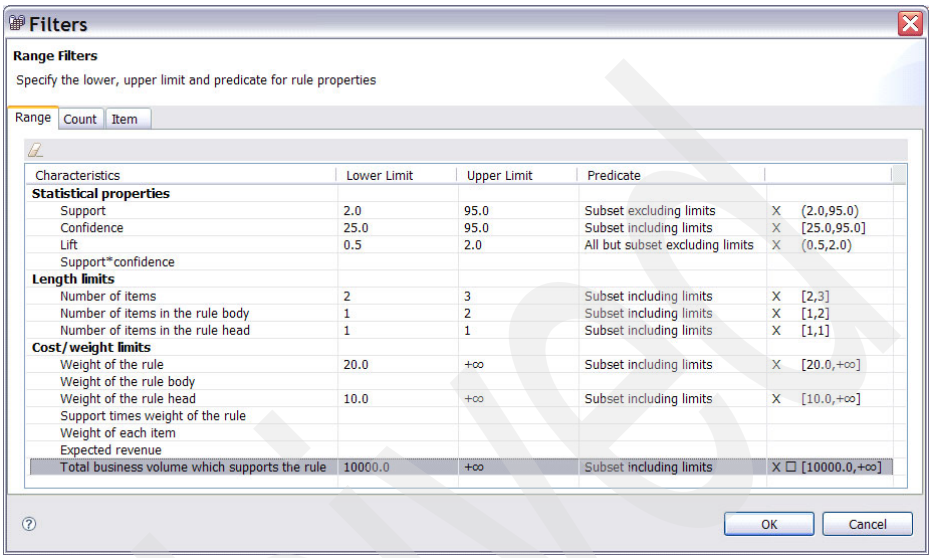


Figure A-49 Filter criteria

- For support and confidence, we have specified an upper limit of 95% in addition to the already existing lower limits of 2% respectively 25%. The idea behind this is that rules with support or confidence of almost 100% are often trivial and well known associations.
- For lift, we specify that we do not want to find rules with lift values between 0.5 and 2. The idea behind this setting is that associations that have lift ≈ 1 are likely to be just co-occurrences of two statistically independent items.
- Concerning the maximum rule length, we specify that we want to have one or two items in the rule body and exactly one item in the rule head.
- Furthermore, we specify that we want to find only those rules whose items' weights (multiplied by the average number of occurrences of each item in the transactions that support the rule) sum up to a total weight of at least 20.
- For the head part of the rule, we specify that the head item's weight, multiplied by the average number of its occurrences in the transactions that support the rule, is at least 10.
- And finally, we specify that we want to find only those rules whose supporting transactions, when all their transaction weights are summed up, represent a total weight (for example, a total business volume) of at least 10000 weight units (for example $).

Three other weight-based filter criteria that we have not used in the above example also require a few words of explanation:

- The criterion 'Support times weight of the rule' is the product of the rule's weight and its absolute support. Hence, this quantity represents the total value or weight of the rule in the training data of the associations model.
- The criterion 'Weight of each item' can be used to exclude particularly “cheap” or particularly “expensive” items from the generated rules. Here, the weight of each item is understood as the product of a single item's average weight and its average multiplicity in the input data's transactions. For example, if one carton of milk costs $0.83 on average, and if on average 2.7 cartons of milk are bought together, then the item 'milk' would still pass the filter 'Weight of each item > $1.'
- The criterion 'Expected revenue' is particularly useful for associations-based customer selection in target marketing or cross-selling scenarios. Expected revenue is defined as:

  ExpectedRevenue = (1-confidence) absoluteBodySupport confidence weightOfHead

  Imagine we are preparing a marketing campaign for a product B and we have detected a rule A ==> B that has a high confidence, say 70%. It is a good idea to offer the product B to all customers that have bought A but not (yet) B. How much total revenue (in weight units) can we expect from this action? We assume that after receiving the offer to buy B, 70% of the contacted customers indeed buy B, as the rule says. The total number of contacted customers is (1-confidence) absoluteSupport(A) because this is exactly the number of customers who have bought A but not B. Combining all this with the average weight of a single product B, we arrive at the formula given above for the expected total revenue from this association-based marketing action.

Rule filters as post-filters

Once an association model has been created, rule filters can be applied on this model in order to create a filtered model that contains a subset of the association rules contained in the original model.

A scenario where this feature can be useful is market basket analysis in a large enterprise, for example, a large retailer. Based on a central large transactions data table, a generic market basket associations model could be built. This model can then be sent to several departments of this enterprise: sortiment planners, shop planners, and marketing people, and everybody can extract those rules from the model that are of interest for his or her concrete use case. This approach has two advantages. First, all those different people do not need access to the original transactions’ data. Second, filtering an associations model is a matter of seconds. Building an associations model on large transaction data is a matter of hours or days.

In DB2 Warehouse V9.1.2 and earlier, using rule filters as post-filter is only possible through the DB2 command line or through the custom SQL operator. In a future DB2 Warehouse version, there will probably be a new post-filter operator which has a window named Rule Filters in its Properties view similar to the Associations operator.

In Example A-4, we show the SQL code that takes an existing model RetailAssocModel and produces a sub-model called RetailAssocFilteredModel in which all rules with lift values between 0.5 and 1.5 are filtered out.

Example: A-4 Rule filters

```
INSERT INTO IDMMX.RULEMODELS( MODELNAME, MODEL )
  SELECT
    'RetailAssocFilteredModel',
     IDMMX.DM_filterRuleModel( MODEL, IDMMX.DM_RuleFilter()
       ..DM_addRangeConstr('lift',0.5,1.5,'notInOpen') )
  FROM IDMMX.RULEMODELS WHERE MODELNAME='RetailAssocModel';
```

A-Priori Associations

There is a large variety of algorithms for detecting associations. All of them have to deal with the challenge of “combinatorical explosion”. That means that in most cases it is not feasible to enumerate all possible combinations of three, four, or more items in order to create each possible association of length 3, 4, or longer. As a consequence, almost all algorithms try to explore the *search space* of possibly interesting associations by stepwise extension. Stepwise extension means adding one additional item to a previously validated shorter association from which we already know that it fulfills all specified filter criteria, or for which it is at least possible that an extension fulfills all filter criteria.

Algorithm

The various algorithms differ in the following ways:

- The order in which the tree of possible association extensions is constructed: depth-first versus breadth-first extension.
- In many use cases, it is not possible to keep all training data records, all already validated and accepted associations, and all currently examined associations candidates (and their parents, from which they derive) in memory at the same time. Therefore, the algorithm has to decide which objects are kept in memory and which ones are stored in dump files on disk or read from a data base whenever needed.

- Which filter criteria can be tested in an early state of the candidate creation and validation process. Sorting out as many candidates as possible as early as possible can dramatically improve the model creation speed.

The A-Priori algorithm, invented by Rakesh Agrawal at IBM Research in the early 1990s, is the “grandfather” of all associations algorithms. A-Priori is a breadth-first candidate expansion algorithm that uses the minimum support criterion and the maximum rule length criterion as central filter criteria. A-Priori keeps large lists of candidate patterns with identical number of items in memory and rereads the training data from the data base whenever a set of candidate patterns is to be validated against the training data. When all candidate patterns of length n have been validated, all candidates that pass the minimum support criterion are combined with every item that is not yet part of the candidate. Thereby, a new candidate list of associations of length n+1 is created that can then again be validated against the training data and that afterwards serves as seed for creating the list of all candidates of length n+2. This is continued until the maximum rule length is reached.

A-Priori makes use of the fact that support is monotonically decreasing with rule length: if an association of length n is below the minimum support threshold, all associations deriving from it are also below the minimum support threshold. The name “A-Priori” has been chosen because the algorithm knows *a priori* which candidates of length n+1 are possibly successful: only those ones that derive from a parent that passes the minimum support barrier. In pseudocode, A-Priori can be formulated as shown in Example A-5.

Example: A-5 A-Priori pseudocode

```
{items} := set of all items with support ≥ minSupport
{parentCandidates} := {items}
{acceptedRules} := {}
n := 1
do {
  {candidates} := {parentCandidates}   {items}
..n := n+1
  perform data scan, determine support({candidates})
  delete candidates with support < minSupport
  {parentCandidates} := {candidates}
  delete candidates which don't pass the other filters
  {acceptedRules} += {candidates}
} until n = maximumRuleLength
```

Parameters and advanced options

A-Priori Associations supports Taxonomies, Name Mappings, and the basic rule filter criteria available through the Mining Settings window of the Associations operator: minimum support, minimum confidence, and maximum rule length.

All other parameters and options described in the previous sections are only supported by DB2 Warehouse SIDE Associations.

Applications of A-Priori Associations

DB2 Warehouse offers A-Priori Associations mainly for backward compatibility reasons. This algorithm was already present in the predecessor products IBM Intelligent Miner™ for Data and IBM DB2 Intelligent Miner Modeling. For DB2 Warehouse users who migrate their existing Intelligent Miner for Data MiningBases or their existing Intelligent Miner Modeling SQL scripts, A-Priori offers an algorithm that exactly reproduces the parameters, results, and runtimes they would have obtained with the predecessor products.

For all other applications of Associations mining, we recommend using the DB2 Warehouse default SIDE algorithm. This algorithm is much faster than A-Priori in many use cases, and unlike A-Priori it supports all rule filter criteria described in “Algorithm” on page 462.

Further reading

- Agrawal, et al, “Mining Associations between Sets of Items in Massive Databases” in Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data, 207-216, 1993.
- Agrawal, et al, “Fast Discovery of Association Rules” in Advances in Knowledge Discovery and Data Mining, AAAI Press, 307-328, 1996.
- Agrawal, et al, “Constraint-Based Rule Mining in Large, Dense Databases” in Proceedings of the 15th International Conference on Data Engineering, 1999.
- Introduction to contingency tables:

  http://en.wikipedia.org/wiki/Contingency_table

SIDE Associations

All associations detection algorithms proposed so far use a step-by-step approach for iteratively traversing the search space of candidate association rules. That means they start with some trivial seed rules and then construct and evaluate child rule candidates by again and again adding one single new item to an existing rule.

Algorithm

The various algorithms differ in:

- How the candidate expansion process is organized (depth-first versus breadth-first expansion)
- Which part of the required information for the expansion and evaluation is kept in memory and which part is read from a data base or recomputed when needed

There are four kinds of required information:

1. The list of all unevaluated candidate rules.
2. The list of evaluated and accepted rules (and item sets).
3. The training data needed for evaluating the candidate rules.
4. Some algorithms also memorize which data records support each accepted parent rule that is currently been used as a seed for creating extended child candidate rules. We call this information the “active records history”. The purpose of storing this active records history is that we only have to read the active transactions (and not all training data) when evaluating the child candidate rules.

The information (2) is normally quite small, but (1), (3), and (4) can be so large that they cannot be stored in memory simultaneously.

In the past, there were two classes of algorithms. The first class, for example the A-Priori algorithm:

- Stores unevaluated candidate rules in memory.
- Does not store the training data. The data are read from the data base whenever they are needed.
- Does not use and store active record histories. Each new candidate rule is evaluated against all training data records.
- Performs breadth-first rule expansion: First all rule candidates with two items are created and evaluated, then all with three, then all with four items, and so on.

The disadvantage of this approach is that it is relatively slow because each candidate rule has to be evaluated against all training data.

The second class of algorithms, for example, the SPAM algorithm:

- Stores preprocessed training data in memory.
- Stores active record histories in memory.

- Performs a depth-first search: One single rule candidate with two items is created and evaluated against the training data. Then the child candidates of this one parent are created and evaluated one by one against the parent's active records, and each child is immediately used as parent for creating and evaluating new children

Figure A-50 shows in which order the depth-first expansion approach creates all possible associations of the five items A, B, C, D, and E. The blue number at each rule represents the creation order.

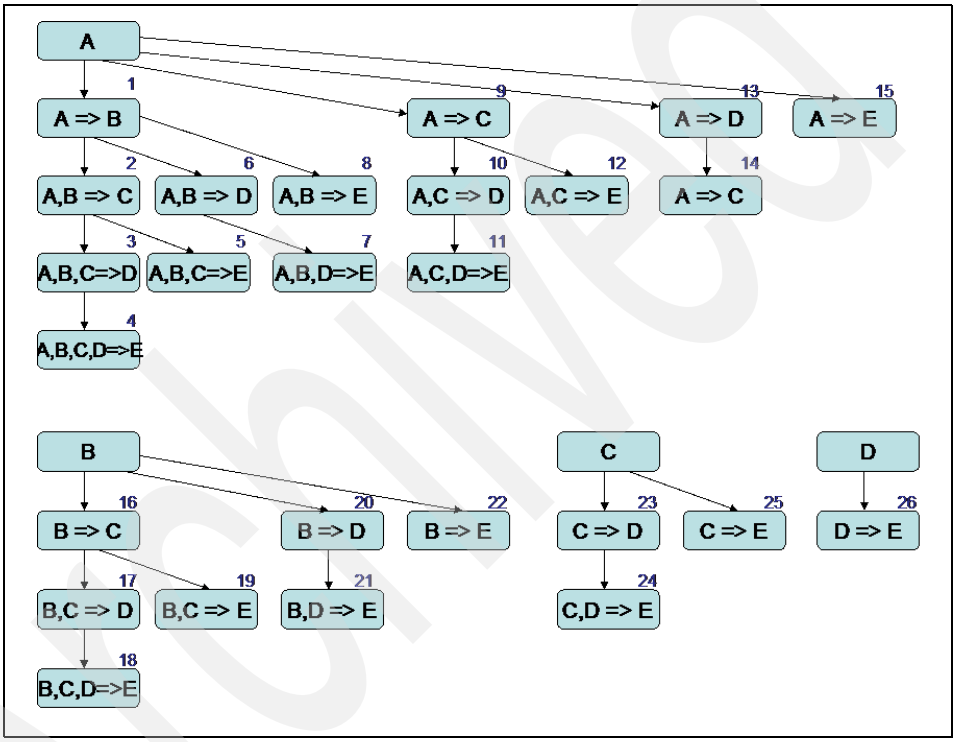


Figure A-50 Expansion of all possible associations

The disadvantage of this approach is that it is memory-demanding and you can run out of memory with large training data. In this case, parts of the training data are dumped to disk and the algorithm becomes very slow, because a depth-first search requires more data scans than a breadth-first search.

The SIDEs algorithm (**SI**multaneous **D**epth-first **E**xpansion) tries to combine the breadth-first and the depth-first approach in order to avoid the disadvantages of the two approaches. This algorithm has been invented and filed for patent in March 2006 by IBM Data Mining Development in Böblingen (Germany). The algorithm consists of four parts:

- A pre-pass over the data that collects statistics on frequent items and frequent item pairs.
- A data preprocessing step that transforms the transactions containing frequent item pairs into a more efficient binary data format. The binary data typically consumes not more than 5%-7% of the original data size. In many cases, this is small enough to be able to keep all binary data in memory.
- The main training step. This step iteratively creates lists of candidate rules, evaluates them against the data, and finds those ones that pass all specified filter criteria. Then new, longer candidate rules are created by adding new items to the accepted candidates of the previous iteration.
- A post-processing step that calculates lift values for item sets and associations, applies lift-based filter criteria, and composes the resulting PMML model.

The pre-pass: determine frequent items and item pairs

This step performs a first full pass over all training data (DB2 table access).

- If there are continuous numeric item columns, sample 10000 data records in order to determine good bin boundaries for discretizing these fields.
- A full pass over all training data (DB2 table access). Collect frequent items and frequent item pairs. Store this item (pair) statistics in an item hash table with 64-bit hashes for the item names.
- If one or more taxonomies have been defined in the mining task, extend the item (pair) statistics to all taxonomy parents of the items found in the training data.
- In the unlikely case of a hash collision, which means if different items have the same hash, redo the pass and recreate the hash table with another hash seed. A hash collision is a rare event that appears on average once in 1019 item names.
- Finalize the pre-pass by throwing away all items and item pairs that are infrequent or that are deactivated by the rule filter.
- Assign each remaining item an integer item ID between 0 and the number of frequent items minus 1.

The data preparation and compression step

This step performs a second pass over the training data (DB2 table access) and transforms the data into a binary representation in which:

- All input data records that belong to 64 consecutive transactions are grouped into one binary data object, called TA-group, using bit fields for attaining compression.
- The TA-group objects are organized in several DataPage objects, each of them having a memory size of between 2 and 64 MB, depending on the total amount of available RAM memory.
- The DataPage objects normally reside in RAM, but if there is not enough memory to store all of them simultaneously, each DataPage object can dump itself to disk and re-fetch itself transparently for the caller when needed again.

After this second data pass, we do not need the item hash table any more. Each item is now described by its 16-bit (or 32-bit) ID. Therefore, we can delete the item and item pair hash table and store the remaining frequent item IDs and their frequent pair information in a simple array over all frequent item IDs, which consumes less memory and allows faster access than the hash table.

Finally, if one or more name mapping tables are given in the mining task, these name mapping tables are read and their content is stored in a name mapping dictionary.

From now on, we do not need data access to DB2.

The Main Training step

In this step, we adopt a third way that tries to use the best features of breadth-first and depth-first rule expansion.

- We try to store all information (preprocessed and compressed and paged training data, candidate associations under evaluation, and active-data-record-histories) in memory as long as possible.
- We put all data in paged and compressed binary objects that can dump themselves to disk when memory is full, and that reload themselves automatically and transparently through fast block-write whenever needed again.
- Depth-first candidate expansion starts from seed candidate lists of the form 'item1 ==> item2' where 'item1' is fixed and item2 varies. This allows you to use the parents' active record history, so that we only have to evaluate the child candidates against a small fraction of all training data.

- But in each expansion and evaluation step of the depth-first approach, we create and evaluate an entire list of up to 32 or 64 or 128 similar candidates simultaneously, similar to the breadth-first approach. The actual number is optimized for the bit-size of the integer registers of the used CPU, thereby enabling fast bit mask operations on the entire set of candidates.

Rule filter constraints

These are applied as early in the training process as possible. This is a key feature for keeping the training times low:

- Negative item constraints, the collected information about item and item pair frequencies, and constraints on the number of items can be applied before or during the construction of new child candidates, before having to validate against the data. Therefore, these filter settings are the most efficient ones and can drastically cut down the training time.
- Minimum support constraints can only be applied after validating against the data, but then they allow you to stop generating further child levels. This is due to the fact that a child candidate can never have larger support than its parent. Therefore, this filter also efficiently reduces training times.
- A maximum lift constraint far below 1, for example 0.02, is also an efficient filter criterion because it implicitly induces a minimum support criterion on the body part of the rule. In our example, the maximum lift threshold of 0.02 induces a minimum absolute support threshold for the body part of the rules of 50.

  This type of constraint is typical for deviation detection scenarios, since a rule that occurs sometimes, but much less frequently than expected, can be interpreted as a deviation or exception.
- The other filters based on support, confidence, lift, or positive item constraints) do not stop the further child creation process if a given candidate fails on them. They are therefore not efficient in cutting down the training time.

Parameters and advanced options

The following advanced options are directly supported by the various sub-windows of the Association operator's Properties view in the DB2 Warehouse DesignStudio.

Group Field

In this box, you can specify one column in the input data as the group field. A group field should only be specified if the input data table has the transactional format, which means if one transaction spans more than one row in this table.

Rule filters

The basic rule filters 'maximum rule length', 'minimum support', and 'minimum confidence' have their own specific input fields in the Mining Settings window. More advanced filter settings can be defined using the Rule Filters window. Details on using rule filters are described in “Rule filters” on page 454.

Name mappings and taxonomies

Details on defining mapped cleartext names for item IDs and for defining item hierarchies or taxonomies can be found in “Name mappings and taxonomies” on page 482.

Weight (cost or price) information

Details on assigning a weight, cost, or price to certain items can be found in “Specifying weights: cost or price” on page 449.

Item formats

Details on defining how the items in the input data are interpreted and in which textual format they appear in the generated association can be found in “Item formats” on page 482.

Field usage types

In the Column Properties window of the Associations operator's Properties view, each available input data field has a default usage type System determined, which means that each of these fields is interpreted as an item field. You can overwrite this behavior for any data field if you click the value **System determined** in the table column Field Usage Type. A menu pops up from which you can select either **Inactive** or **Weight**. Inactive fields are simply ignored for the mining. Weight fields are assumed to contain price or weight information for one of the item fields. The mapping between item fields and associated weight fields is established by selecting a weight field's name in the Weight Column entry of the item field's record in the Column Properties window.

Field types

The Column Properties view of the Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields, whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one separate textual item, and no discretization or binning is performed for this field.

For DB2 Warehouse versions higher then 9.1.2, an additional field type multi_categorical can be chosen in the Field Types field of the Column Properties window. If this field type is selected for a field, DB2 Warehouse SIDE Associations expects to find XML strings of the form:

<item>$text_1$</item><item>$text_2$</item>…<item>$text_n$</item>

Here, $text_1$ to $text_n$ are arbitrary categorical values. Each of them will be interpreted as one separate field value. The stop words <item> and </item> are not interpreted as parts of the field's values; they only serve to separate one value from the following one. Optionally, you can wrap the string <item>$text_1$ … $text_n$</item> into a framing <itemset>…</itemset> root tag, and you can also prepend an XML header string such as '<?xml version="1.0" encoding="xxx" ?>. This format has the advantage of being a well-formed XML document.

The format multi_categorical is particularly useful in connection with text analysis. For example, the analysis of a patient's medical history might have detected the keywords apnoe, smoker, or obesity. Then this text analysis result can be cast into one single multi-valued field as:

<item>apnoe</item><item>smoker</item><item>obesity</item>.

The following advanced options do not have a directly corresponding input field in the Associations operator's Properties view. You have to type the API command into the general purpose Optional Parameters field in the Mining Settings window. Figure A-51 shows this for the algorithm parameter NumBins, together with the power option -buf.

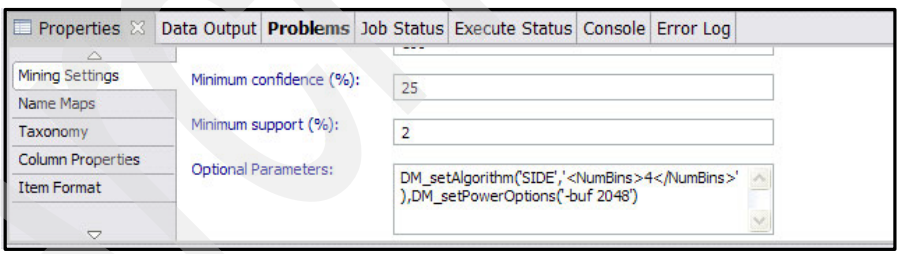


Figure A-51 Advanced options entry

Discretization mode

Details on modifying the discretization of numeric item fields can be found in “Specifying the discretization mode for numeric fields” on page 452.

Performance Tuning

The algorithm parameter NumRecordsForNewItems can be used for speeding up the training process and reducing memory requirements for the cost of a loss of completeness guarantee for the created model.

Unlike other mining tasks, for example, clustering, associations mining tasks are deterministic. That means once the input data and the rule filter criteria are determined, every associations algorithm should produce the same set of rules. That is indeed the default behavior of the associations algorithms: They guarantee that they exactly solve the deterministic associations task, and not a single rule is missing in the created model.

Using the parameter NumRecordsForNewItems, the guarantee to not miss a single rule is abandoned. The parameters tells the algorithm to stop searching for new, previously unseen items after reading a certain number of data records in the statistics collecting pre-pass. The idea is that an item that never occurs in the first 5% or 10% of the data is very unlikely to pass the minimum support threshold at the end. The effect of stopping the search for new items early in the pre-scan is that the item hash table remains much smaller and the pre-scan runs faster. The exact syntax of the command is (with x being an absolute number of records):

```
DM_setAlgorithm('SIDE','<NumRecordsForNewItems>x
  </NumRecordsForNewItems>')
```

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within 'clause' must be surrounded by two single quotes. Here is an example:

```
DM_setWhereClause('GENDER=''m''').
```

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field 'fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

SIDE Associations supports the following power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the mining kernel. As a rule of thumb, SIDE associations runs with optimum speed if the available memory for the mining kernel has at least about 6% of the size of the input data.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.

Content of an Associations model

DB2 Warehouse SIDE Associations models are stored as PMML <AssociationModel> in the repository table IDMMX.RULEMODELS. They contain the following information.

Model quality

An associations mining task is a deterministic task, and for each task there exists only one correct associations model. If a SIDE Associations task has enough memory resources to build that model, the model's quality is 1. But if the mining kernel runs out of memory and has to stop the statistics pre-scan after x% of all data records, or if the parameter <NumRecordsForNewItems> has been set to x% of all data records, then the model quality is 0.01•x.

Items

For each item that is contained in one of the detected associations, the model contains the original item value, the mapped value (if a name mapping exists), the item's weight statistics (if weight info exists), and an item ID running from 1 to the number of items. The weight of each item accounts for the average multiplicity of this item per transaction. For example, if customers by on average 2.3 cartons of milk at $1 each per purchase, the mean weight if item 'milk' is 2.3.

Taxonomy

If a taxonomy has been defined, the model contains all parent-child pairs in which the child is an item that appears in at least one association of the model.

Item sets

Each item set that appears as the rule body or the rule head of an association rule of the model is contained with its items, its support, its lift, its weight statistics (if weight info exists), and its itemset ID running from 1 to the number of itemsets.

Associations

Each association of the model is contained with its body and head itemset IDs, its support, its confidence, its lift, its weight statistics (if weight info exists), and its rule ID running from 1 to the number of rules.

Methods and operators

In this section, we discuss methods and operators for extracting model content into DB2 tables.

Associations Extractor operator

The mining flow editor of the Design Studio contains an operator called Associations Extractor. This operator reads an association model and creates three tables.

The first table, RULE, contains the following columns:

- ID (integer): ID of the association rule.
- BODYID (integer): A generated identifier to allow joining the resulting table with a table generated by DM_getRuleBodies.
- HEAD (varchar(1024)): Original head item before name mappings.
- HEADNAME (varchar(1024)): Contains the head item of the rule after name mapping (if any).
- LENGTH (smallint): The number of items in head and body.
- BODYTEXT (varchar(4096)): A concatenation of the names of all body items separated by the term 'AND'.
- SUPPORT (real): Support of the association rule. A value between 0 and 1.
- CONFIDENCE (real): Confidence of the association rule. A value between 0 and 1.
- LIFT (real): Lift of the association rule.

The second table, RULEBODY, contains the following columns:

- BODYID (integer): Contains the ID of the rule body. Each value in BODYID appears in n rows, where n is the size of the item set. There is a distinct entry in BODYID for each item set appearing in a rule of the rule model.
- ITEM (varchar(1024)): Original item before name mappings.
- ITEMNAME (varchar(1024)): Contains the item of the rule body after name mapping (if any).

The third table, ITEMSET contains the following columns:

- ITEMSETID (integer): Contains the ID of the item set.
- ITEM (varchar(1024)): Original item before name mappings.
- ITEMNAME (varchar(1024)): Contains the item of the itemset after name mapping (if any).
- SUPPORT (real): The support of the item set. The support value is a number between 0 and 1. It represents the ratio between the number of transactions that support (contain) the item set and the total number of transactions.

Quality Extractor operator

The Quality Extractor operator writes the association model's quality Q into a DB2 table.

More precisely, the Quality Extractor, which can also be used with any other DB2 Warehouse mining model, writes four numbers: model quality Q and three detailed model quality numbers that are only available in classification and regression models: reliability, accuracy, and ranking quality. When applied to an association model, these columns in the output will contain SQL NULL.

Fields Extractor

The Fields Extractor operator writes a table containing the following information for each active mining field of the association model:

- COLNAME: Name of the data column in the training data table
- FIELDNAME: By default, this is equal to COLNAME; contains the field alias if an alias has been defined using the command DM_setFldAlias.
- MININGTYPE: Set to 0 if the field is categorical, set to 1 if it is numeric.
- IMPORTANCE: This column always contains SQL NULL in association models.

SQL API commands and Custom SQL operator

The DB2 Warehouse mining SQL API offers some more model extractor methods for which no specific graphical mining flow operator exists. These commands can be used within a mining flow by means of the Custom SQL operator. Figure A-52 shows how the API command DM_getRulesW is called within a Custom SQL operator. The screen capture shows the Properties view of a Custom SQL operator, which has been drawn to the DesignStudio' Mining Flow Editor canvas and whose input port has been connected to the table source RULE_TABLE.

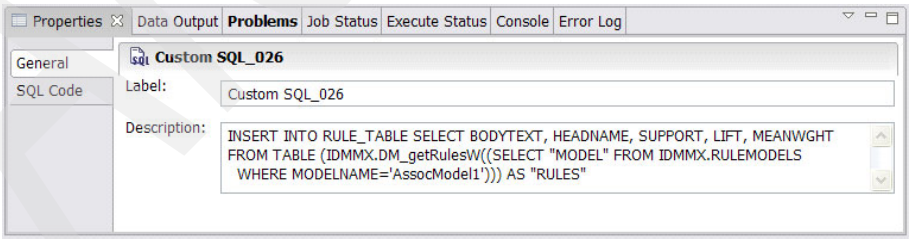


Figure A-52 Custom SQL Operator

This operator writes the rule properties BODYTEXT, HEADNAME, SUPPORT, LIFT, and MEANWGHT of each rule from the model AssocModel1 into the table RULE_TABLE.

The following model introspection commands for association models, which means objects of type IDMMX.DM_RuleModel, are available:

- IDMMX.DM_expRuleModel(ruleModel): Returns the PMML model as a flat XML file in the form of a CLOB.
- IDMMX.DM_getRuleMdlName(ruleModel): Returns the model's name.
- IDMMX.DM_getRuleMdlType(ruleModel): Returns the type of the rule model: 'association'.
- IDMMX.DM_getMdlQuality(ruleModel): Returns the model quality Q.
- IDMMX.DM_getNumRules(ruleModel): Returns the number of association rules contained in the model.
- IDMMX.DM_getNumTransacts(ruleModel): Returns the number of transactions in the input data on which the model was trained.
- IDMMX.DM_getGroup(ruleModel): Returns the name of the group field.
- IDMMX.DM_filterRuleModel(ruleModel,filter): Returns that part of 'ruleModel' which passes the filter 'filter'.
- IDMMX.DM_getItemsets(ruleModel): This command is the API equivalent of the third output port of the Associations Extractor operator. It returns a 5-column table (ITEMSETID, ITEM, ITEMNAME, SUPPORT, and LIFT) in which each row describes one item of an item set. The column entries ITEMSETID, SUPPORT, and LIFT are repeated for all items of the item set.
- IDMMX.DM_getItemsetsW(ruleModel): This command is an extended version of DM_getItemsets that also extracts the item set's weight statistics. The command returns a 13-column table (ITEMSETID, ITEM, ITEMNAME, SUPPORT, LIFT, MEANWGHT, STDDEVWGHT, MINWGHT, MAXWGHT, MEANTAWGHT, STDDEVTAWGHT, MINTAWGHT, and MAXTAWGHT) in which each row describes one item of an item set. The column entries ITEMSETID, SUPPORT, and LIFT and the eight weight entries are repeated for all items of the item set. The first four weight columns describe the aggregated weight of all items in the item set. The last four weight columns describe the average weight of the transactions that contain the item set.
- IDMMX.DM_getRuleBodies(ruleModel): This command is the API equivalent of the second output port of the Associations Extractor operator. It returns a 3-column table (BODYID, ITEM, and ITEMNAME) in which each row describes one item of an item set. The column entry BODYID is repeated for all items of the item set.
- IDMMX.DM_getRules(ruleModel): This command is the API equivalent of the first output port of the Associations Extractor operator. It returns a 9-column table (ID, LENGTH, BODYID, BODYTEXT, HEAD, HEADNAME, SUPPORT, CONFIDENCE, and LIFT) in which each row describes one association rule.

- IDMMX.DM_getRulesW(ruleModel): This command is an extended version of DM_getRules that also extracts the rules' weight statistics. The command returns a 17-column table (ID, LENGTH, BODYID, BODYTEXT, HEAD, HEADNAME, SUPPORT, CONFIDENCE, LIFT, MEANWGHT, STDDEVWGHT, MINWGHT, MAXWGHT, MEANTAWGHT, STDDEVTAWGHT, MINTAWGHT, and MAXTAWGHT) in which each row describes one association rule. The first four weight columns describe the aggregated weight of all items in the rule. The last four weight columns describe the average weight of the transactions that contain the association rule.
- IDMMX.DM_getFields(clusModel): Returns a four-column table (COLNAME, FIELDNAME, MININGTYPE, and IMPORTANCE) that contains for each active field in the model the name of the data column to which it refers, its name, its mining type (categorical or numeric), and its importance, a value between 0 and 1.

Applications of SIDE Associations

Due to its flexible rule filters, SIDE Associations can be applied to a wide variety of use cases from almost any industry, for example:

- Market basket analysis: Find articles that are often sold together. Use this insight for store planning and target marketing.
- Assortment planning and profitability analysis.
- Quality assurance: Find combinations of components or production parameters that produce above-average failure rates.
- Warranty claim analysis: Detect component combinations and other factors that are overrepresented in warranty claims.
- Medical studies: Verify the effectiveness of medical treatment; detect risk factors for a disease.
- Deviation detection and fraud detection.
- Data quality monitoring: Find data records that seem to contain wrong or inconsistent entries.

Further Reading

- Agrawal, et al, “Fast Discovery of Association Rules”, Advances in Knowledge Discovery and Data Mining, AAAI Press, 307-328, 1996.

Scoring an Association model

Once an Association model has been created and visually checked, you might want to apply it to either the training data or to new data. Applying or scoring an Association model means something different than scoring a Clustering, Classification or Regression model. In the latter mining functions, scoring a data record against a model means adding one or more scalar values to that record, for example, a cluster ID, or a predicted value and a confidence. Scoring a table of length N means creating a new table of the same length and with the same primary key as the input table, but with one or more additional columns.

With an Association model, scoring means linking all rules in the model with all transactions in the data that support the rule. Therefore, when you apply the Scorer operator to an Association model and table of transactions, you receive a new table in which the combination of transaction ID (group ID) and rule ID forms a combined key.

In Figure A-53 on page 476, we have connected a transactional data table from a retailer and an Association model containing frequent article combinations to a Scorer operator. The data is in transactional (or pivoted) format. Column ITEMID contains the purchased items, and column TRANSID is the group column that contains the purchase IDs.

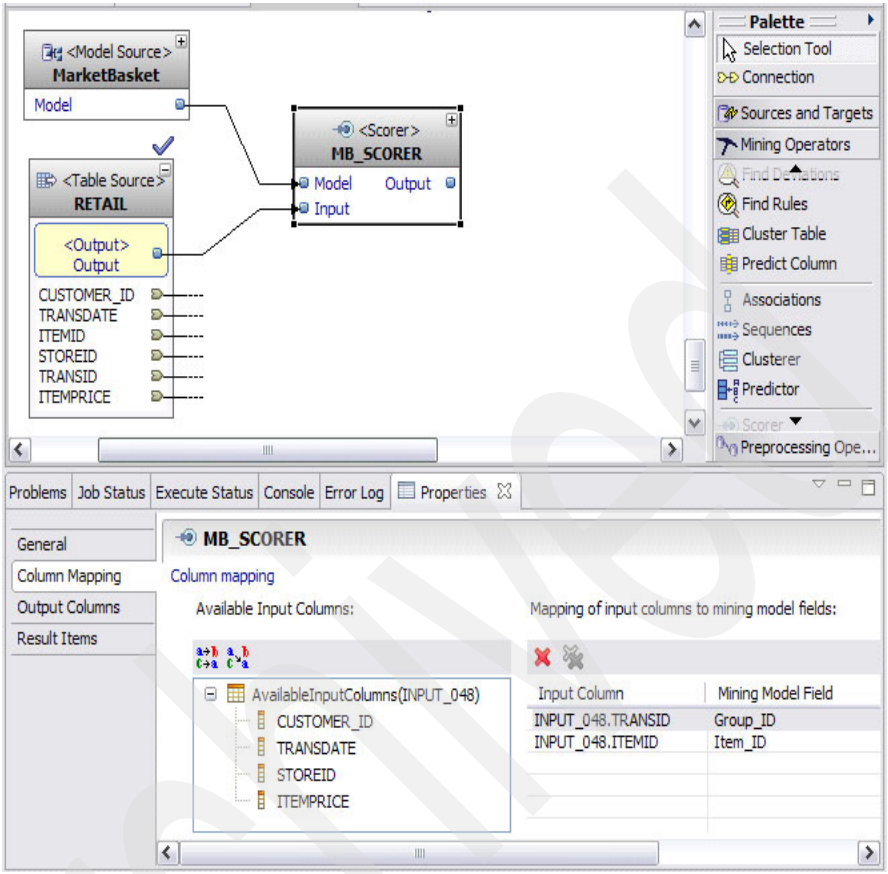


Figure A-53 Connecting transactional data to a Scorer operator

In the Properties view of the Scorer operator, we have manually established that mapping of input columns names to their function within the Association model: We have dragged the column name TRANSID into the first row of the table on the left and the column name ITEMID into the second row.

Now we have to provide a table into which the scoring results will be written. In many cases, a result table of exactly matching column format does not yet exist. Instead of manually creating a new table, you can right-click the output port of the Scorer operator and select the menu item **Create suitable table**. A wizard pops up in which you are presented with a set of default settings for the new table. In Figure A-54, we have accepted all the default settings.

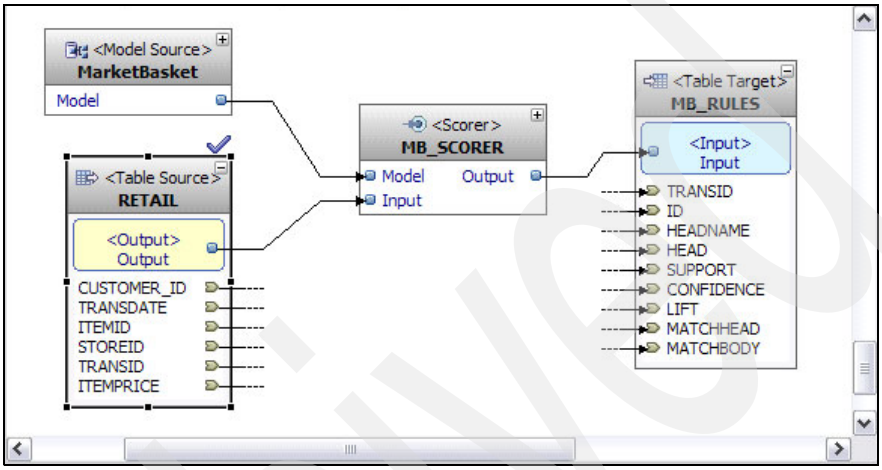


Figure A-54 Default table settings from Scorer operator

Scoring result measures

In the generated table, the columns TRANSID and ID contain pairs of transaction ID plus rule ID that serve as a unique key in the new table. HEAD contains the item value that forms the right hand side of the rule with ID ID .. HEADNAME is the cleartext mapped name of the head item if such a name mapping exists. SUPPORT, LIFT, and CONFIDENCE are the statistical properties of the rule. MATCHHEAD and MATCHBODY are two Boolean 0/1 columns that indicate whether the transaction TRANSID contains the body of the rule ID, or its head, or both.

Limitations

The DB2 Warehouse Scorer operator for association models has two limitations:

- It only works on input data in transactional (pivoted) format.
- It is unable to handle taxonomy relations. This means that if a transaction contains 'cream' and 'toy car', the scorer is unable to link that transaction to the rule 'milk products ==> toys'.

Sequences

The Sequences mining function finds typical time-ordered sequences of items in your data.

Introduction and notations

An item can be an arbitrary “atomic” event, article, fact, or component in the data that carries a time stamp. A prerequisite for finding sequences between these atomic items is that a two-fold grouping and assignment of several of the items to one comprising and superior entity exists. First, one can interpret all items with identical time stamps as one item set or transaction (TA). Second, there must be still another grouping entity that labels several transactions as belonging to one entity instance, often called a transaction group (TAG). A sequence or sequential pattern is a time-ordered series of item sets that appear in this form in many transaction groups. We write this as

$\{itemset_1\}$ >>> $\{itemset_2\}$ >>> … >>> $\{itemset_n\}$.

If the last time step of the sequence is considered as the step between the body and the head of a rule, we interpret the sequence as sequence rule and we write it as:

$\{itemset_1\}$ >>> … >>> $\{itemset_{n-1}\}$ ==> $\{itemset_n\}$.

The left hand side of the sequence rule is called the rule body or antecedent, and the right hand side the rule head or consequent.

Table A-9 lists typical use cases of mining for sequences.

Table A-9 Banking services example

Industry	Use case	Item grouping (time stamp)	TA-Grouping	Typical body item	Typical head items
Retail	Upselling	Time stamp of a bill	Customer ID	A purchased article	Another purchased article
Mfg.	Root cause analysis	Production step, or date+time	Product ID (for example, vehicle)	Problem error ID	Problem error ID
Medicine	Med study evaluation	Treatment step or date	Patient ID or Test Person	Single treatment info	Medical impact

Typical sequence rules for the three use cases listed in the table are:

- {computer} >>> {printer} ==> {ink cartridge}
- {v-belt broken} ==> {no battery power}
- {Hypomed Forte 5 ml} >>> {Placebon Extra 2ml} ==> {patient cured}

Sequence rule properties

A sequence rule can be characterized by the following properties:

- The contained items (in the rule body, in the rule head, and in the entire rule)
- Categories of the contained items. Often, an additional hierarchy or taxonomy for the items is known. For example, the items 'milk' and 'baby food' might belong to the category 'food', 'diapers' might belong to the category 'non-food'. 'axles=17-B' and 'engine=AX-Turbo 2.3' might be members of the category 'component', 'production_chain=3' of category 'production condition', and 'delay > 2 hours' of category 'error state'. Hence, the second sample rule can be characterized by the fact that its body contains components or production conditions, and its head an error state.
- The support of the rule. Absolute support is defined as the total number of entities (transaction groups) for which the rule holds. Support or relative support is the fraction of all transaction groups for which the rule holds.
- The confidence of the rule. Confidence is defined as

  ```
  confidence := support(body==>head) / support(body)
  ```

- The lift of the rule. Lift is defined as:
  - lift := support(body==>head) / (2 • support(body)• support(head)).
  - lift>1 means that the rule 'body==>head' appears more frequently than expected assuming that 'body' and 'head' are statistically independent.

  The factor of 2 in the denominator stems from time ordering, or more precisely, from the fact that in the case of statistical independence of body and head we would expect:
  - support(body==>head) = support(head==>body)
  - support(body==>head) + support(head==>body) = support(body) support(head)
- The weight (cost,price) of the rule. Often, an economical value measure exists for each item, for example, its price or cost. If this is the case, the aggregated weight (cost or price) of a sequence or sequence rule can be defined as the sum of all item weights multiplied with their multiplicities. For example, if those customers that support the sequence rule '{newborn baby food} ==> {baby toys}' buy on average 4.3 portions of newborn baby food at $1 each and at a later time 1.3 baby toys at $5 each, the average weight of the sequence rule is $10.80.

- The weight (cost,price) of the transaction groups that support the rule (TAG weight). If an economical value measure exists for each item, a second weight property for a sequence rule in addition to its average weight can be given: the average weight of the transaction groups that support the rule. If, for example, those customers who first purchase {newborn baby food} and at a later time {baby toys} also purchased other items at an total average price of $354.13 during the time period represented in the available data, then the TAG weight of the rule '{newborn baby food} ==> {baby toys}' is $10.80+$354.13 = $365.96.

Data formats

In this section, we discuss the data formats for sequencing.

The transactional or pivoted data format

Often, the input data for sequence rule mining are available in a three-column format in which one column is the TA-group or sequence field and contains the transaction group IDs. The second field is the group or time stamp field containing transaction IDs or time stamps. The third field is the item field and contains items. For example, an input table for market basket analysis could be as depicted in Table A-10.

Table A-10 Market basket analysis example

CUSTOMER_ID	TRANS_DATE	ITEM
000001	2007-02-21	Diet Coke
000001	2007-02-21	Toast
000001	2007-02-26	Milk
000001	2007-02-26	Cheddar
000002	2007-02-13	Beer
000002	2007-02-13	Dark Bread

Here, the first purchase of customer '000001' contains the articles 'Diet Coke' and 'Toast', the second one the items 'Milk' and 'Cheddar' and 'Beer'. The first purchase of customer 000002 comprises 'Beer' and 'Dark Bread'. For this data format, you have to specify the group column (in the example, TRANS_DATE) and the sequence column (in the example, CUSTOMER_ID) in the 'Properties' view of the Sequences operator, as depicted in Figure A-55.

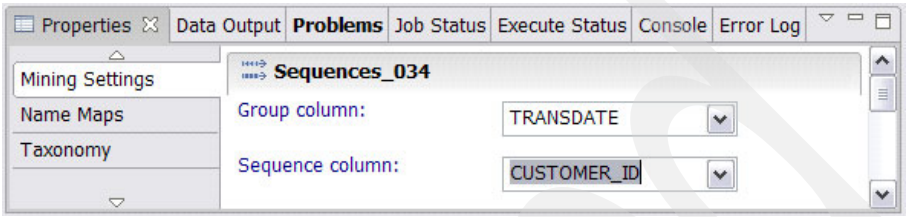


Figure A-55 Sequence operator: Properties view

The unpivoted or broad data format

In other use cases, the data are available in a format in which each row in the table corresponds to one transaction (unpivoted data format). Such a table contains more than one item field or at least one set-valued item field. This data format can, but need not, contain a group or time stamp field; if no group field exists, each transaction within one transaction group is assigned an artificial integer time stamp running from 1 to the number of transactions in the group. Table A-11 shows the equivalent to our first sample table in unpivoted format. The table contains one set-valued item field.

Table A-11 Market basket analysis example: unpivoted

CUSTOMER_ID	TRANS_DATE	ITEM
000001	2007-02-21	<item>diet coke</item><item>toast</item>
000001	2007-02-26	<item>Cheddar</item><item>beer</item>
000002	2007-02-13	<item>beer</item><item>dark bread</item>

By default, all input table columns that are not flagged as sequence column or group column are interpreted as item columns. Therefore, each column whose content should not appear in the generated sequences has to be switched to **inactive** in the Column Properties window of the Sequence operator's Properties page.

Item formats

Independently of the global data table format (transactional with group field or broad without group field), each single item column can have differently formatted content. Furthermore, for each item column, the way the contained items should appear in the generated sequences can be modified. You can adjust both aspects of item formats to your data and your needs. There is a coarse-grained way of adjusting item formats by means of a graphical GUI panel in the Sequences operator's Properties view, and a fine-grained way using textual commands in the Optional Parameters text field.

Since item format handling in the Sequences operator is identical to item format handling in the Associations operator, we refer to “Item formats” on page 441 for more details.

Name mappings and taxonomies

Often the value of the sequence rule mining result can be increased if together with the main transactions data source two auxiliary data sources are used.

Name mappings define cleartext names for the items in the main transaction table if the latter in their original form are unintuitive numeric IDs. This helps to create better understandable association rules.

Taxonomies define hierarchies or categories for the items that are contained in the transaction data. This information can be used for powerful filter criteria and for creating more general sequences that cover not single items but entire item categories.

For further details on specifying Name Mappings and Taxonomies, please refer to “Name mappings and taxonomies” on page 447.

Specifying weights: cost or price

In most cases, you have a certain economic motivation for running a mining for associations. It is therefore desirable that the detected sequences can directly express the economic value that they represent. With DB2 Warehouse Sequences mining, you can do this if you specify so-called weight information for the involved items. The weight of an item can be any numeric measure representing the importance of an item, such as its price, cost, physical weight, size, area, duration, and so on. In business applications, cost or price are the most frequently used weights.

For further details on specifying weights, please refer to “Specifying weights: cost or price” on page 449.

Specifying the discretization mode for numeric fields

Whenever the input data for sequence rule mining contains numeric fields, in particular floating point fields, these data must be discretized, because otherwise each single numeric value would appear only once or a small number of times in the data, and no such specific value would reach the required minimum support threshold. Sequences automatically perform this discretization, called binning. You can modify the default binning method and adjust it to your needs.

For further details on specifying weights, please refer to “Specifying the discretization mode for numeric fields” on page 452.

Rule filters

A frequent challenge in mining for sequence rules is that too many rules are found, and it is difficult to detect the really interesting ones within thousands or tens of thousands of other rules that are either trivial or not helpful for the specific use case. Therefore, it is often necessary to define a set of rule filter criteria before starting the mining run. These criteria help both to keep the resulting set of sequences manageable and to speed up the training process.

Sequences mining supports the same very flexible and powerful rule filters as Associations mining. For further details on rule filters, please refer to “Rule filters” on page 454.

SIDE sequences

In this section, we discuss side sequences.

Algorithm

All sequence detection algorithms proposed so far use a step-by-step approach for iteratively traversing the search space of candidate sequences. That means they start with some trivial seed rules and then construct and evaluate child rule candidates by again and again adding one single new item to an existing rule.

The various algorithms differ in:

- How the candidate expansion process is organized (depth-first versus breadth-first expansion)
- Which part of the required information for the expansion and evaluation is kept in memory and which part is read from a data base or recomputed when needed

There are four kinds of required information:

1. The list of all unevaluated candidate rules.
2. The list of evaluated and accepted rules (and item sets).
3. The training data needed for evaluating the candidate rules.
4. Some algorithms also memorize which data records support each accepted parent rule that is currently been used as a seed for creating extended child candidate rules. We call this information the active records history. The purpose of storing this active records history is that we only have to read the active transactions (and not all training data) when evaluating the child candidate rules.

The information (2) is normally quite small, but (1), (3), and (4) can be so large that they cannot be stored in memory simultaneously.

In the past, there were two classes of algorithms. The first class, for example, the A-Priori algorithm:

- Stores unevaluated candidate rules in memory.
- Does not store the training data. The data are read from the data base whenever needed.
- Does not use and store active record histories. Each new candidate rule is evaluated against all training data records.
- Performs breadth-first rule expansion: First, all rule candidates with two items are created and evaluated, then all with three, then all with four items, and so on.

The disadvantage of this approach is that it is relatively slow because each candidate rule has to be evaluated against all training data.

The second class of algorithms, for example, the SPAM algorithm:

- Stores preprocessed training data in memory.
- Stores active record histories in memory.
- Performs a depth-first search: One single rule candidate with two items is created and evaluated against the training data. Then the child candidates of this one parent are created and evaluated one by one against the parent's active records, and each child is immediately used as parent for creating and evaluating new children

The disadvantage of this approach is that it is memory-demanding and can run out of memory for large training data. In this case, parts of the training data are dumped to disk and the algorithm becomes very slow because depth-first search requires more data scans than breadth-first search.

The SIDE algorithm (SImultaneous Depth-first Expansion) tries to combine the breadth-first and the depth-first approach in order to avoid the disadvantages of the two approaches. This algorithm has been invented and filed for patent in March 2006 by IBM Data Mining Development in Böblingen (Germany). The algorithm consists of four parts:

- A pre-pass over the data that collects statistics on frequent items and frequent item pairs.
- A data preprocessing step that transforms the transaction groups containing frequent item pairs into a more efficient binary data format. The binary data typically consume not more than 5%-7% of the original data size. In many cases, this is small enough to be able to keep all binary data in memory.
- The main training step. This step iteratively creates lists of candidate rules, evaluates them against the data, and finds those ones that pass all specified filter criteria. Then new, longer candidate rules are created by adding new items to the accepted candidates of the previous iteration.
- A post-processing step that calculates lift values for item sets and sequences, applies lift-based filter criteria, and composes the resulting PMML model.

The pre-pass: determine frequent items and item pairs

This step performs a first full pass over all training data (DB2 table access).

- If there are continuous numeric item columns, sample 10000 data records in order to determine good bin boundaries for discretizing these fields.
- Full pass over all training data (DB2 table access). Collect frequent items and frequent equal-time and different-time item pairs. Store this item (pair) statistics in an item hash table with 64-bit hashes for the item names.
- If one or more taxonomies have been defined in the mining task, extend the item (pair) statistics to all taxonomy parents of the items found in the training data.
- In the unlikely case of a hash collision, that means if different items have the same hash, redo the pass and recreate the hash table with another hash seed. Note: A hash collision is a rare event that appears on average once in 1019 item names.
- Finalize the pre-pass by throwing away all items and item pairs that are infrequent or that are deactivated by the rule filter.
- Assign each remaining item an integer item ID between 0 and the number of frequent items minus 1.

The data preparation and compression step

This step performs a second pass over the training data (DB2 table access) and transforms the data into a binary representation in which:

- All input data records that belong to one transaction group are grouped into one binary data object, called TA-group, using bit fields for attaining compression.
- The TA-group objects are organized in several DataPage objects, each of them having a memory size of between 2 and 64 MB, depending on the total amount of available RAM memory.
- The DataPage objects normally reside in RAM, but if there is not enough memory to store all of them simultaneously, each DataPage object can dump itself to disk and re-fetch itself transparently for the caller when needed again.

After this second data pass, we do not need the item hash table any more. Each item is now described by its 16-bit (or 32-bit) ID. Therefore, we can delete the item and item pair hash table and store the remaining frequent item IDs and their frequent pair information in a simple array over all frequent item IDs, which consumes less memory and allows faster access than the hash table.

Finally, if one or more name mapping tables are given in the mining task, these name mapping tables are read and their content is stored in a name mapping dictionary.

From now on, we do not need any more data access to DB2.

The main training step

In this step, we adopt a third way, which tries to use the best features of breadth-first and depth-first rule expansion.

- We try to store all information (preprocessed and compressed and paged training data, candidate sequences under evaluation, and active-data-record-histories) in memory as long as possible.
- We put all data in paged and compressed binary objects that can dump themselves to disk when memory is full, and which reload themselves automatically and transparently through fast block-write whenever needed again.
- Depth-first candidate expansion, starting from seed candidate lists of the form 'item1 ==> item2', where 'item1' is fixed and item2 varies. This allows using the parents' active record history so that we only have to evaluate the child candidates against a small fraction of all training data.

- But in each expansion and evaluation step of the depth-first approach, we create and evaluate an entire list of up to 32 or 64 or 128 similar candidates simultaneously, similar to the breadth-first approach. The actual number is optimized for the bit-size of the integer registers of the used CPU, thereby enabling fast bit mask operations on the entire set of candidates.

Rule filter constraints are applied as early in the training process as possible. This is a key feature for keeping the training times low:

- Negative item constraints, the collected information about item and item pair frequencies, and constraints on the number of items can be applied before or during the construction of new child candidates, before having to validate against the data. Therefore, these filter settings are the most efficient ones and can drastically cut down the training time.
- Minimum support constraints can only be applied after validating against the data, but then they allow you to stop generating further child levels. This is due to the fact that a child candidate can never have larger support than its parent. Therefore, this filter also efficiently reduces training times.
- A maximum lift constraint far below 1, for example 0.02, is also an efficient filter criterion because it implicitly induces a minimum support criterion on the body part of the rule. In our example, the maximum lift threshold of 0.02 induces a minimum absolute support threshold for the body part of the rules of 50.

  This type of constraint is typical for deviation detection scenarios, since a rule which occurs sometimes, but much less frequently than expected, can be interpreted as a deviation or exception.
- The other filters based on support, confidence, lift, or positive item constraints) do not stop the further child creation process if a given candidate fails on them. They are therefore not efficient in cutting down the training time.

Parameters and advanced options

The following advanced options are directly supported by the various sub-window of the Sequences operator's Properties view in the DesignStudio.

Group field and sequence field

In these two select boxes, you can specify one column in the input data as the group or time stamp field and another column as the transaction-group or sequence field. A group field need not be specified if the input data table has the transactional format, which means if one transaction spans more than one row in this table. A sequence field must always be specified.

Rule filters

The basic rule filters “maximum rule length”, “minimum support”, and “minimum confidence” have their own specific input fields in the Mining Settings window. More advanced filter settings can be defined using the Rule Filters window. Details on using rule filters are described in “Item formats” on page 482.

The only difference for rule filter handling for associations mining is that for sequences, there are three additional range-based filter criteria:

- Number of itemsets (which is equal to the number of time steps plus 1).
- Total elapsed time between first and last item set of the sequence.
- Elapsed time between adjacent item sets of the sequence.

Name mappings and taxonomies

Details on defining mapped cleartext names for item IDs and for defining item hierarchies or taxonomies can be found in “Name mappings and taxonomies” on page 482.

Weight (cost or price) information

Details on assigning a weight, cost, or price to certain items can be found in “Specifying weights: cost or price” on page 482.

Item formats

Details on defining how the items in the input data are interpreted and in which textual format they appear in the generated sequences can be found in “Item formats” on page 482.

Field usage types

In the Column Properties window of the Associations operator's Properties view, each available input data field has a default usage type of System determined, which means that each of these fields is interpreted as an item field. You can overwrite this behavior for any data field if you click the value **System determined** in the table column Field Usage Type. A menu pops up from which you can select either **Inactive** or **Weight**. Inactive fields are simply ignored for the mining. Weight fields are assumed to contain price or weight information for one of the item fields. The mapping between item fields and associated weight fields is established by selecting a weight field's name in the Weight Column entry of the item field's record in the Column Properties window.

Field types

The Column Properties view of the Properties view also contains a column called Field Type, which can take the values System determined or categorical.

System determined means that fields of numeric SQL type (integer, floating point, or date or time types) are considered as numeric fields whereas fields with all other SQL types are considered as non-numeric categorical fields.

Setting the Field Type value to categorical for a numeric field has the effect that each single field value in the training data is considered as one separate textual item, and no discretization or binning is performed for this field.

The following advanced options do not have a directly corresponding input field in the Associations operator's Properties view. You have to type the API command into the general purpose Optional Parameters field in the Mining Settings window. Figure A-56 shows this for the algorithm parameter NumBins, together with the power option -buf.

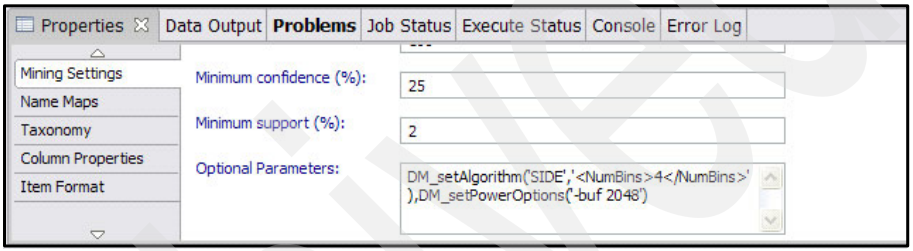


Figure A-56 Associations operator's Properties view

Discretization mode

Details on modifying the discretization of numeric item fields can be found in "Specifying the discretization mode for numeric fields" on page 483.

Performance tuning

The algorithm parameter NumRecordsForNewItems can be used for speeding up the training process and reducing memory requirements at the cost of a loss of the completeness guarantee for the created model.

Unlike other mining tasks, for example, a clustering, sequence rule mining tasks are deterministic. That means that once the input data and the rule filter criteria are determined, every sequence detection algorithm should produce the same set of rules. And that is indeed the default behavior of the Sequences algorithm: It guarantees that it exactly solves the deterministic sequence mining task, and not a single sequence rule is missing in the created model.

Using the parameter NumRecordsForNewItems, this guarantee to not miss a single rule is abandoned. The parameters tells the algorithm to stop searching for new, previously unseen items after reading a certain number of data records in the pre-pass statistics collecting. The idea is that an item that never occurs in the first 5% or 10% of the data is very unlikely to pass the minimum support threshold at the end. The effect of stopping the search for new items early in the pre-scan is that the item hash table remains much smaller and the pre-scan runs faster. The exact syntax of the command is (with x being an absolute number of records):

```
DM_setAlgorithm('SIDE','<NumRecordsForNewItems>x
  </NumRecordsForNewItems>')
```

Internal precision of time stamp values

By default, time stamp differences of the transactions within a transaction group are stored in a compressed 16-bit format in the binary data objects that have been created from the input data. The consequence of this storage format is that time differences can only be expressed with a granularity of 1/65536 of the maximum encountered time distance within a transaction group in the training data. In some situations, this granularity results in visible rounding errors in the time step statistics that are written into the generated model. At the cost of almost doubling the size of the binary data objects, you can enlarge the internal storage format of time stamps and item IDs to 32 bit using the command:

```
DM_setAlgorithm('SIDE','<ItemIDBits>32</ItemIDBits>')
```

Where clause

The command DM_setWhereClause('clause') specifies a SQL WHERE clause, acting as data sampling condition, which is to be applied on the training data. 'clause' contains the WHERE condition without the leading keyword WHERE. String constants within 'clause' must be surrounded by two single quotes. For example:

```
DM_setWhereClause('GENDER=''m''').
```

Field alias

The command DM_setFldAlias('fldName','alias') defines an alias name for the input data field ‘fldName'. Subsequent commands having a field name argument should use the alias instead of the original column name to refer to this field.

Power options

SIDE Sequences supports the following power options as described in Appendix B, “Power options” on page 503:

- -buf grants more memory to the mining kernel. As a rule of thumb, SIDE sequences runs with optimum speed if the available memory for the mining kernel has at least about 6% of the size of the input data.
- -IDM_WORKING_DIRECTORY redirects log and temporary dump outputs.

Content of an sequence model

DB2 Warehouse SIDE Sequence models are stored as PMML <SequenceModel> in the repository table IDMMX.RULEMODELS. They contain the following information.

Model quality

A sequence rule mining task is a deterministic task, and for each task there exists only one correct sequence model. If SIDE Sequences has enough memory resources to build that model, the model's quality is 1. But if the mining kernel runs out of memory and has to stop the statistics pre-scan after x% of all data records, or if the parameter <NumRecordsForNewItems> has been set to x% of all data records, then the model quality is 0.01•x.

Items

For each item that is contained in one of the detected sequences, the model contains the original item value, the mapped value (if a name mapping exists), the item's weight statistics (if weight info exists), and an item ID running from 1 to the number of items. The weight of each item accounts for the average multiplicity of this item per transaction. For example, if customers buy on average 2.3 cartons of milk at $1 each per purchase, the mean weight if item milk is 2.3.

Taxonomy

If a taxonomy has been defined, the model contains all parent-child pairs in which the child is an item that appears in at least one sequence of the model.

Item sets

Each item set that appears in at least one sequence of the model is contained with its items, its support, its weight statistics (if weight info exists), and its itemset ID running from 1 to the number of item sets.

The support of item sets in sequences models is calculated based on transaction groups, not based on transactions as in Associations models.

Sequences

Each item set that appears in at least one sequence of the model is contained with its items, its support, its weight statistics (if weight info exists), and its itemset ID running from 1 to the number of item sets.

Each sequence of the model is contained with its itemset IDs, the time step statistics between the item sets, the time step statistics for the entire sequence from first to last item set, the sequence's support, its weight statistics (if weight info exists), and its sequence ID running from 1 to the number of sequence rules.

The time step statistics is calculated from the training data in the following steps:

- Every transaction group that supports the sequence is traversed from the lowest to the highest group value (time stamp). The first combination of item sets that supports the entire sequence and that matches the filter constraints of the mining task is detected.
- The maximum difference of the group values (time stamps) in the detected list of item sets is calculated. The average of these maximum group-value differences is stored as average time difference between the beginning and the end of the sequence.
- All differences between adjacent group values (time stamps) in the detected list of item sets are calculated. The averages of these group-value differences are stored as average time differences between adjacent item sets of the sequence.

Sequence rules

Each sequence rule of the model is contained with its body and head sequence IDs, the time step statistics between body and head, the rule's support, its confidence, its lift, its weight statistics (if weight info exists), and its rule ID running from 1 to the number of sequence rules. Each sequence rule corresponds one to one to a sequence of the model. The sequence rule is just the sequence; the last time step is interpreted as the separation between the rule body and the rule head.

The time step statistics is calculated from the training data in the following steps:

- Every transaction group that supports the sequence is traversed from the lowest to the highest group value (time stamp). The first combination of item sets that supports the entire sequence and that matches the filter constraints of the mining task is detected.
- The difference between the largest and the second largest of the group values (time stamps) in the detected list of item sets is calculated.
- The average of these group-value differences is stored as average time difference between the body and the head of the sequence rule.

Auxiliary sequences

The sequence rules reference the sequence IDs of their body sequence and their head sequence. But there is no guarantee that each body or head sequence of an existing sequence is also a valid sequence that passes all filter criteria. Therefore, the sequence model often contains auxiliary sequences that did not pass the filter constraints but that are needed to represent body and head of a certain sequence rule. These auxiliary sequences carry the IDs number of rules + 1 and higher. They never contain weight statistics, and they are marked by an XML comment line in the PMML model.

Methods and operators

In this section, we discuss the methods and operators for extracting model content into DB2 tables.

Sequences Extractor operator

The mining flow editor of the Design Studio contains an operator called 'Sequences Extractor'. This operator reads a sequence model and creates three tables.

The first table, SEQRULES, contains the following columns:

- RULEID (integer): ID of the sequence rule.
- BODYSEQID (integer): ID of the sequence that forms the body or antecedent part of the sequence rule. An integer ID that runs from 1 to the number of sequences in the model. It allows joining the resulting table with a table generated by DM_getSeqDetails in order to access detailed information concerning the body sequence.
- HEADSETID (integer): ID of the item set that forms the head or consequent part of the sequence rule. It allows joining the resulting table with a table generated by DM_getItemsets in order to access detail info concerning the head item set.
- BODYTEXT (varchar(4096)): The body or antecedent part of the sequence rule in textual form. The items in this text are represented by their mapped names. Items that form a single item set are concatenated by +. Adjacent item sets of the sequence are concatenated by the symbol >>>. This symbol indicates a time step.
- HEADSETTEXT (varchar(4096)): The head or consequent part of the sequence rule in textual form. The items in this text are represented by their mapped names. Items that form a single item set are concatenated by +. Adjacent item sets of the sequence are concatenated by the symbol >>>. This symbol indicates a time step.
- LENGTH (smallint): The number of item sets in the sequence rule.

- NBOFITEMS (smallint): The number of items in the sequence rule.
- SUPPORT (double): Support of the sequence rule. A value between 0 and 1.
- CONFIDENCE (double): Confidence of the sequence rule. A value between 0 and 1.
- LIFT (double): Lift of the sequence rule. A value larger than 0.
- MEANTIMEDIFF (double): Mean time step between body and head of the sequence rule.
- STDDEVTIMEDIFF (double): Standard deviation of the time step between body and head of the sequence rule.

The second table, SEQUENCES, contains the following columns:

- SEQUENCEID (integer): Contains the ID of the sequence.
- LENGTH (smallint): The number of item sets in the sequence rule.
- NBOFITEMS (smallint): The number of items in the sequence rule.
- SUPPORT (double): Support of the sequence. A value between 0 and 1
- MEANTIMEDIFF (double): Mean time step between the first and the last item set of the sequence.
- STDDEVTIMEDIFF (double): Standard deviation of the time step between the first and the last item set of the sequence.
- SEQTEXT (varchar(4096)): Sequence in textual form. The items in this text are represented by their mapped names. Items that form a single item set are concatenated by +. Adjacent item sets of the sequence are concatenated by the symbol >>>. This symbol indicates a time step.

The third table, SEQDETAILS, contains the following columns:

- SEQUENCEID (integer): Contains the ID of the sequence. Each value in SEQUENCEID appears in n rows, where n is the number of item sets in the sequence. There is a distinct row for each item set appearing in the current sequence.
- POSI (integer): The position of the current item set within the sequence. An integer that runs from 1 to the number of item sets in the sequence.
- POSI (integer): The position of the current item set within the sequence. An integer that runs from 1 to the number of item sets in the sequence.
- SETID (integer): The itemset ID of the current item set. An integer between 1 and the number of item sets in the sequence model.
- SETTEXT (varchar(4096)): The current item set in textual form. The items in this text are represented by their mapped names. The items in the item set are concatenated by +.

- MEANTIMEDIFF (double): Mean time step between the first and the last item set of the sequence.
- STDDEVTIMEDIFF (double): Standard deviation of the time step between the first and the last item set of the sequence.

The fourth table, ITEMSET contains the following columns:

- ITEMSETID (integer): Contains the ID of the item set.
- ITEM (varchar(1024)): Original item before name mappings.
- ITEMNAME (varchar(1024)): Contains the item of the itemset after name mapping (if any).
- SUPPORT (real): The support of the item set. The support value is a number between 0 and 1. It represents the ratio between the number of transaction groups that support (contain) the item set and the total number of transaction groups

Quality Extractor operator

The Quality Extractor operator writes the sequence model's quality Q into a DB2 table. More precisely, the Quality Extractor, which can also be used with any other DB2 Warehouse mining model, writes four numbers: model quality Q and three detailed model quality numbers that are only available in classification and regression models: reliability, accuracy, and ranking quality. When applied to a sequence model, these columns in the output will contain SQL NULL.

Fields Extractor

The Fields Extractor operator writes a table containing the following information for each active mining field of the sequence model:

- COLNAME: Name of the data column in the training data table
- FIELDNAME: By default, this is equal to COLNAME; contains the field alias if an alias has been defined using the command DM_setFldAlias.
- MININGTYPE: 0 if the field is categorical, 1 if it is numeric.
- IMPORTANCE: This column always contains SQL NULL in sequence models.

SQL API commands, and Custom SQL operator

The DB2 Warehouse mining SQL API offers some more model extractor methods for which no specific graphical mining flow operator exists. These commands can be used within a mining flow by means of the Custom SQL operator. Figure A-57 shows how the API command DM_getSeqRulesW is called within a Custom SQL operator. Figure A-57 shows the Properties view of a Custom SQL operator, which has been drawn to the DesignStudio's Mining Flow Editor canvas and whose input port has been connected to the table source 'SEQRULE_TABLE'.

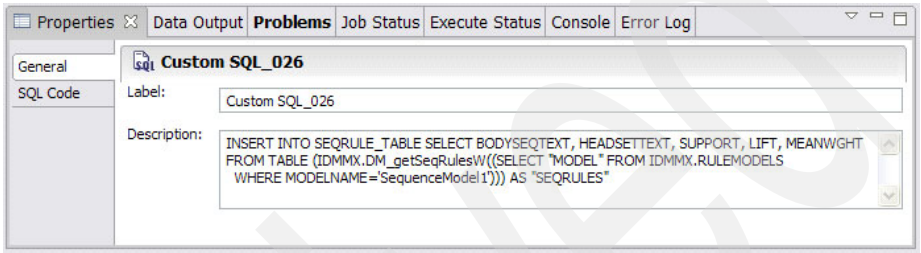


Figure A-57 Custom SQL operator

This operator writes the rule properties BODYTEXT, HEADNAME, SUPPORT, LIFT, and MEANWGHT of each rule from the model 'SequenceModel1' into the table SEQRULE_TABLE.

The following model introspection commands for sequence models, which means objects of type IDMMX.DM_RuleModel, are available:

- IDMMX.DM_expRuleModel(ruleModel): Returns the PMML model as a flat XML file in the form of a CLOB.
- IDMMX.DM_getRuleMdlName(ruleModel): Returns the model's name.
- IDMMX.DM_getRuleMdlType(ruleModel): Returns the type of the rule model: 'association'.
- IDMMX.DM_getMdlQuality(ruleModel): Returns the model quality Q.
- IDMMX.DM_getNumRules(ruleModel): Returns the number of sequence rules contained in the model.
- IDMMX.DM_getNumTransacts(ruleModel): Returns the number of transactions in the input data on which the model was trained.
- IDMMX.DM_getNumTAGroups(ruleModel): Returns the number of transaction groups in the input data on which the model was trained.
- IDMMX.DM_getGroup(ruleModel): Returns the name of the time stamp or group field.

- IDMMX.DM_getSequence(ruleModel): Returns the name of the transaction-group or sequence field.
- IDMMX.DM_filterRuleModel(ruleModel,filter): Returns that part of 'ruleModel' that passes the filter 'filter'.
- IDMMX.DM_getItemsets(ruleModel): This command is the API equivalent of the fourth output port of the Sequences Extractor operator. It returns a 5-column table (ITEMSETID, ITEM, ITEMNAME, SUPPORT, and LIFT) in which each row describes one item of an item set. The column entries ITEMSETID and SUPPORT are repeated for all items of the item set. The column LIFT always contains SQL NULL for sequence models.
- IDMMX.DM_getItemsetsW(ruleModel): This command is an extended version of DM_getItemsets that also extracts the item set's weight statistics. The command returns a 13-column table (ITEMSETID, ITEM, ITEMNAME, SUPPORT, LIFT, MEANWGHT, STDDEVWGHT, MINWGHT, MAXWGHT, MEANTAWGHT, STDDEVTAWGHT, MINTAWGHT, and MAXTAWGHT) in which each row describes one item of an item set. The column entries ITEMSETID, SUPPORT, and LIFT and the eight weight entries are repeated for all items of the item set. The first four weight columns describe the aggregated weight of all items in the item set. The last four weight columns describe the average weight of the transactions that contain the item set.
- IDMMX.DM_getSeqDetails(ruleModel): This command is the API equivalent of the third output port of the Sequences Extractor operator. It returns a 6-column table (SEQUENCEID, POSI, SETID, SETTEXT, MEANTIMEDIFF, and STDDEVTIMEDIFF) in which each row describes one item set of a sequence. The column entry SEQUENCEYID is repeated for all items of the item set.
- IDMMX.DM_getSeqDetailsW(ruleModel): This command is an extended version of DM_getSeqDetails, which also extracts the rules' weight statistics. The command returns a 14-column table (SEQUENCEID, POSI, SETID, SETTEXT, MEANTIMEDIFF, STDDEVTIMEDIFF, MEANWGHT, STDDEVWGHT, MINWGHT, MAXWGHT, MEANTAGWGHT, STDDEVTAGWGHT, MINTAGWGHT, and MAXTAGWGHT) in which each row describes one item set of a sequence in the model. The first four weight columns describe the aggregated weight of all items in the item set. The last four weight columns describe the average weight of the transaction groups that contain the item set. The column entry SEQUENCEYID is repeated for all item sets of the sequence.

- IDMMX.DM_getSeqRules(ruleModel): This command is the API equivalent of the first output port of the Sequences Extractor operator. It returns a 12-column table (RULEID, BODYSEQID, HEADSETID, BODYSEQTEXT, HEADSETTEXT, LENGTH, NBOFITEMS, SUPPORT, CONFIDENCE, LIFT, MEANTIMEDIFF, and STDDEVTIMEDIFF) in which each row describes one sequence rule.
- IDMMX.DM_getSeqRulesW(ruleModel): This command is an extended version of DM_getRules that also extracts the rules' weight statistics. The command returns a 20-column table (RULEID, BODYSEQID, HEADSETID, BODYSEQTEXT, HEADSETTEXT, LENGTH, NBOFITEMS, SUPPORT, CONFIDENCE, LIFT, MEANTIMEDIFF, STDDEVTIMEDIFF, MEANWGHT, STDDEVWGHT, MINWGHT, MAXWGHT, MEANTAGWGHT, STDDEVTAGWGHT, MINTAGWGHT, and MAXTAGWGHT) in which each row describes one sequence rule. The first four weight columns describe the aggregated weight of all items in the rule. The last four weight columns describe the average weight of the transaction groups that contain the sequence rule.
- IDMMX.DM_getSequences(ruleModel): This command is the API equivalent of the second output port of the Sequences Extractor operator. It returns a 7-column table (SEQUENCEID, LENGTH, NBOFITEMS, SUPPORT, MEANTIMEDIFF, STDDEVTIMEDIFF, and SEQTEXT) in which each row describes one sequence.
- IDMMX.DM_getSequencesW(ruleModel): This command is an extended version of DM_getSequences that also extracts the sequences' weight statistics. The command returns a 15-column table (SEQUENCEID, LENGTH, NBOFITEMS, SUPPORT, MEANTIMEDIFF, STDDEVTIMEDIFF, SEQTEXT, MEANWGHT, STDDEVWGHT, MINWGHT, MAXWGHT, MEANTAGWGHT, STDDEVTAGWGHT, MINTAGWGHT, and MAXTAGWGHT) in which each row describes one sequence. The first four weight columns describe the aggregated weight of all items in the sequence. The last four weight columns describe the average weight of the transaction groups which contain the sequence.
- IDMMX.DM_getFields(clusModel): Returns a four-column table (COLNAME, FIELDNAME, MININGTYPE, and IMPORTANCE) that contains for each active field in the model the name of the data column to which it refers, its name, its mining type (categorical or numeric), and its importance, a value between 0 and 1.

Applications of SIDE Associations

Due to its flexible rule filters, SIDE Sequences can be applied to a wide variety of use cases from almost any industry. SIDE Sequences can be used whenever transaction, production, or other log data is available that contain time stamps or another ordering criterion. Sample applications are:

- Market basket analysis: Find articles that are often bought after other articles. Use this insight for target marketing (up-selling).
- Quality assurance: Find time-ordered combinations of components or production parameters that produce above-average failure rates.
- Warranty claim analysis: Detect typical patterns in warranty claim histories.
- Medical studies: Verify the effectiveness of medical treatment; detect typical patient histories for a disease.
- Deviation detection and fraud detection on time-ordered data.

Further Reading

- *Effective Mining Sequential Patterns by Last Position Induction*, found at:

  http://www.ieice.org/iss/de/DEWS/DEWS2005/procs/papers/6C-o2.pdf

Scoring a sequence model

Once a sequence model has been created and visually checked, you might want to apply it to either the training data or to new data. Applying or scoring a sequence model means something different than scoring a clustering, classification or regression model.

In the latter mining functions, scoring a data record against a model means adding one or more scalar values to that record, for example, a cluster ID, or a predicted value and a confidence. Scoring a table of length N means creating a new table of the same length and with the same primary key as the input table, but with one or more additional columns.

With a sequence model, scoring means linking all sequence rules in the model with all transaction groups in the data that support the rule. Therefore, when you apply the Scorer operator to a sequence model and a table of transactions, you receive a new table in which the combination of transaction group ID and sequence rule ID forms a combined key.

In Figure A-58, we have connected a transactional data table from a retailer and a sequence model containing typical sequences of purchased articles to a Scorer operator. The data is in transactional (or pivoted) format. Column ITEMID contains the purchased items, column TRANSDATE is the group (or time stamp) column, and column CUSTOMER_ID is the transaction group (or sequence) column that contains the customer IDs.

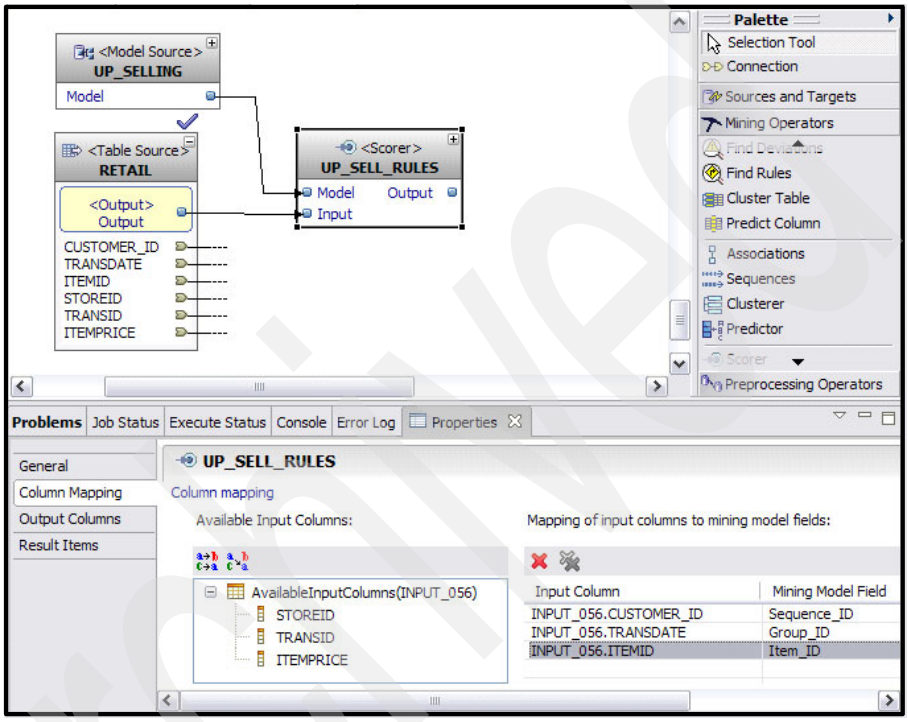


Figure A-58 Properties view of Scorer

We have manually established mapping of input columns names to their function within the sequence model: We have dragged the column name CUSTOMER_ID into the first row of the table on the left, the column name TRANSDATE into the second, and ITEMID into the third row.

Now we have to provide a table into which the scoring results will be written. In many cases, a result table of exactly matching column format does not yet exist. Instead of manually creating a new table, you can right-click the output port of the Scorer operator and select the menu item **Create suitable table**. A wizard pops up in which you are presented with a set of default settings for the new table. As depicted in Figure A-59, we have accepted all default settings.

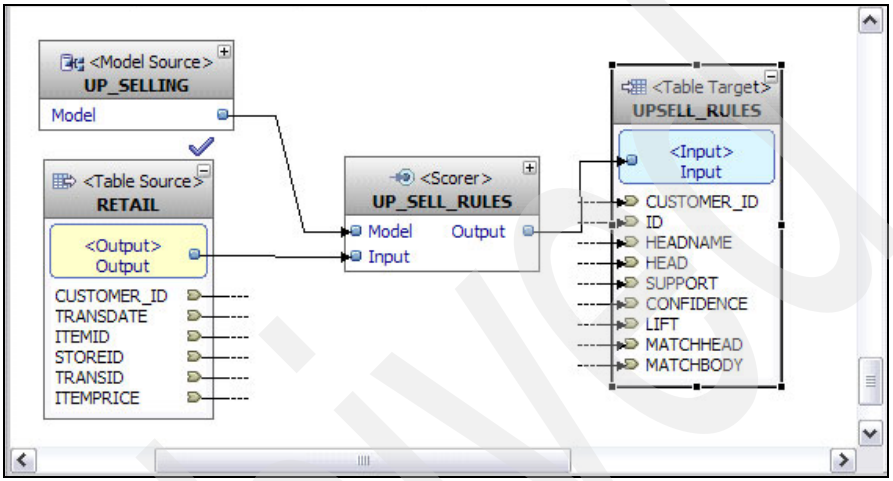


Figure A-59 Scoring Results Table

Scoring result measures

In the generated table, the columns CUSTOMER_ID and ID contain pairs of transaction group IDs plus sequence rule IDs that serve as a unique key in the new table. HEAD contains the item values that form the right hand side of the rule with ID ID. HEADNAME is the string of cleartext mapped names of the head items if such name mappings exist. SUPPORT, LIFT, and CONFIDENCE are the statistical properties of the rule. MATCHHEAD and MATCHBODY are two Boolean 0/1 columns that indicate whether the transaction group CUSTOMER_ID contains the body of the rule ID, or its head, or both.

Limitations

The Scorer operator for sequence models has two limitations:

- It only works on input data in transactional (pivoted) format.
- It is unable to handle taxonomy relations. That means if a transaction contains 'cream' and 'toy car', the scorer is unable to link that transaction to the rule 'milk products ==> toys'.

B

Power options

In addition to algorithm-specific advanced parameters, there are settings and restrictions that are valid for all data mining processes started from DB2W, independently of the mining function or mining algorithm. Those restrictions affect, for example, the amount of hard disk and RAM resources that are made available for the mining tasks, or the way in which statistical information about the training data is collected and stored.

For modifying the default behavior with respect to these restrictions, DB2W offers the concept of power options and the command DM_setPowerOptions, which can be issued both from the DB2W SQL application programming interfaces and from the graphical Design Studio.

In this appendix, we describe these power options.

Input syntax

The general format for setting one or more power options is:

```
DM_setPowerOptions('-option1 value1 [-option2 value2 …]')
```

Each new call of DM_setPowerOptions overwrites the results of earlier calls. Therefore, in general, each mining flow or mining task should contain only one call of DM_setPowerOptions. Otherwise, the result might not be the desired one, because all options set in earlier calls to DM_setPowerOptions are ignored.

Design Studio

In the graphical Design Studio, each data mining operator within a mining flow has a free text field called Optional Parameters on the Mining Settings page of its Properties view. Into this text field, the command DM_setPowerOptions can be typed or pasted. If more than one optional parameter command is used, the different commands must be separated by commas.

Figure B-1 shows how the power option -buf 1024 is specified for a Demographic Clustering mining flow. This power option increases the available amount of RAM for the execution of the mining flow from the default value of 512 MB to 1024 MB. Together with the power option, an algorithm-specific advanced parameter is set that reduces the importance or weight factor of the input field AGE from the default value 1 to 0.5.

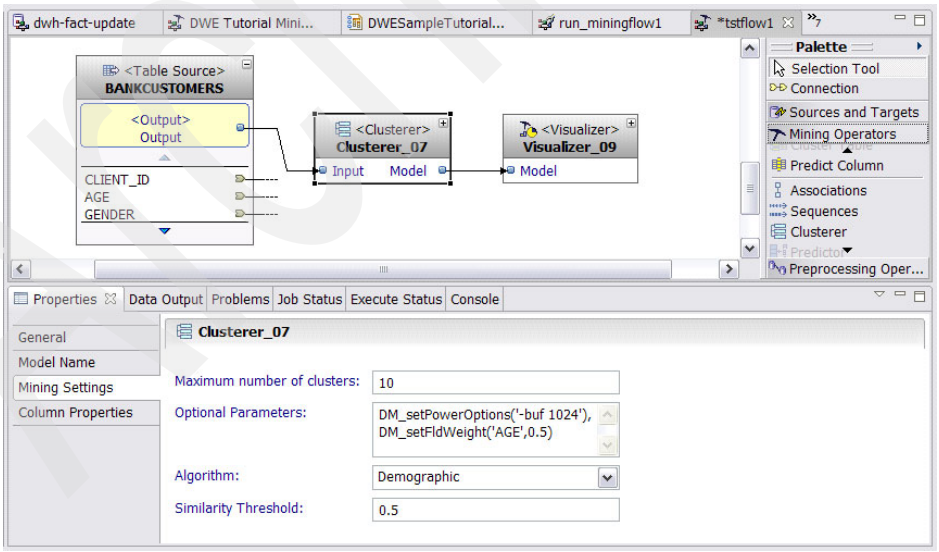


Figure B-1 Specifying power options

EasyMining SQL API

Using the EasyMining SQL API, the command corresponding to the Clusterer operator shown above would be:

```
CALL IDMMX.BuildClusModel('CustomerSegmentation','BANKCUSTOMERS',
      'DM_setAlgorithm(''Demographic''),DM_setMaxNumClus(10),
       DM_setPowerOptions(''-buf 1024''), DM_setFldWeight
       (''AGE'',0.5)')
```

Each EasyMining API command has an optional last argument of type VARCHAR into which the caller can pass an arbitrary combination of algorithm-specific parameters and power options, separated by commas. This optional argument string is shown in bold in the command printed above. Within the options string, embedded string arguments such as '-buf 1024' have to be enclosed in two repeated single quotes.

Low level SQL API (SQL/MM)

In the low level SQL API, which implements the SQL/MM (part 6) standard, the Clustering mining flow shown above could be expressed as depicted in Example B-1.

Example: B-1 Clustering mining flow example

```
INSERT INTO IDMMX.CLUSTASKS ("ID", "TASK" )
WITH "MYDATA" ( "MYDATACOL" ) AS
  (VALUES(IDMMX.DM_MiningData()..DM_defMiningData('BANKCUSTOMERS')))
SELECT
  'CustomerSegmentationTask',
  IDMMX.DM_ClusBldTask()..DM_defClusBldTask(
    "MYDATACOL",
    IDMMX.DM_ClusSettings()
      ..DM_useClusDataSpec("MYDATACOL"..DM_genDataSpec())
      ..DM_setAlgorithm('Demographic')
      ..DM_setMaxNumClus(10)
      ..DM_setPowerOptions('-buf 1024')
      ..DM_setFldWeight('AGE',0.5)
  )
FROM "MYDATA";

CALL IDMMX.DM_buildClusModelCmd(
  'IDMMX.CLUSTASKS', 'TASK', 'ID', 'CustomerSegmentationTask',
  'IDMMX.CLUSTERMODELS', 'MODEL', 'MODELNAME', 'CustomerSegmentation'
);
```

Power options reference

In this section, we describe the power options.

-IDM_WORKING_DIRECTORY

This power option replaces the default working directory for DB2W Data Mining error files and temporary dump files by a user-defined directory path. The default working directory is /tmp on Linux and UNIX systems and C:\Documents and Settings\Administrator\Local Settings\Temp on Windows.

Some DB2W mining kernels, in particular the Tree Classification, Transform Regression, SIDE Associations, and SIDE Sequential Patterns modeling kernels, temporarily write a compressed binary representation of parts of the training data into the working directory during the model training process. These temporary binary data files can become as large as about 5-10% of the original training data. Therefore, when working on very large data, the default working directory might not have sufficiently free hard disk space, and redirecting the output to another directory might be necessary.

Here is an example:

```
DM_setPowerOptions('-IDM_WORKING_DIRECTORY D:\temp')
```

-buf

This power option redefines the maximum amount of memory that is made available to each mining run. Per default, in DB2W Version 9 on Linux and UNIX systems, each mining run may consume up to 1 GB of memory, and on Windows systems up to 512 MB. The argument after option -buf is the granted amount of memory in MB. In large real-world use cases and input data, in particular the mining algorithms SIDE Associations, SIDE Sequential Patterns, and Tree Classification can sometimes be accelerated significantly if they are granted more than the default memory threshold.

Here is an example:

```
DM_setPowerOptions('-buf 2048')
```

-IDM_MAX_DISCR_COUNT

This power option redefines the maximum number of different values of a discrete input field that will be considered by the DB2W mining algorithms and shown with their statistical measures in the resulting model. By default, 100 different discrete values are considered. If a discrete input field contains more than 100 different values, all but the first 100 different values are treated as invalid values by the DB2W data mining algorithms. Reducing this number might significantly reduce both the runtime for generating a mining model and the

resulting model's size, but it might also reduce the accuracy or clarity of the model.

Here is an example:

```
DM_setPowerOptions('-IDM_MAX_DISCR_COUNT 20')
```

-IDM_MAX_DISCR_NUM_COUNT

This is a variant of -IDM_MAX_DISCR_COUNT that only affects the discrete numeric fields of the input data. If both -IDM_MAX_DISCR_COUNT and -IDM_MAX_DISCR_NUM_COUNT are set, the latter overwrites -IDM_MAX_DISCR_COUNT for the discrete numeric fields.

Here is an example:

```
-IDM_MAX_DISCR_CAT_COUNT
```

This is a variant of -IDM_MAX_DISCR_COUNT that only affects the categorical fields of the input data. If both -IDM_MAX_DISCR_COUNT and -IDM_MAX_DISCR_CAT_COUNT are set, the latter overwrites -IDM_MAX_DISCR_COUNT for the categorical fields.

Glossary

Access Control List (ACL). The list of principals that have explicit permission (to publish, to subscribe to, and to request persistent delivery of a publication message) against a topic in the topic tree. The ACLs define the implementation of topic-based security.

Additive Measure. Measure of a fact that can be added across all dimensions.

Aggregate. Pre-calculated and pre-stored summaries, kept in the data warehouse to improve query performance

Aggregation. An attribute level transformation that reduces the level of detail of available data. For example, having a Total Quantity by Category of Items rather than the individual quantity of each item in the category.

Analytic. An application or capability that performs some analysis on a set of data.

Application Programming Interface. An interface provided by a software product that enables programs to request services.

Associations. Data mining analysis that finds links or associations among the data records of single transactions.

Associative entity. An entity created to resolve a many-to-many relationship into two one-to-many relationships.

Asynchronous Messaging. A method of communication between programs in which a program places a message on a message queue, then proceeds with its own processing without waiting for a reply to its message.

Attribute. A characteristic of an entity, such as a field in a dimension table.

BLOB. Binary Large Object, a block of bytes of data (for example, the body of a message) that has no discernible meaning, but is treated as one solid entity that cannot be interpreted.

Business subject area. A particular function or area within an enterprise whose processes and activities can be described by a defined set of data elements.

Candidate key. One of multiple possible keys for an entity.

Cardinality. The number of elements in a mathematical set or group.

Clustering. Data mining method of grouping data by how similar they are based on attributes of interest.

Commit. An operation that applies all the changes made during the current unit of recovery or unit of work. After the operation is complete, a new unit of recovery or unit of work begins.

Compensation. The ability of DB2 to process SQL that is not supported by a data source on the data from that data source.

Composite Key. A key in a fact table that is the concatenation of the foreign keys in the dimension tables.

Computer. A device that accepts information (in the form of digitalized data) and manipulates it for some result based on a program or sequence of instructions on how the data is to be processed.

Configuration. The collection of brokers, their execution groups, the message flows and sets that are assigned to them, and the topics and associated access control specifications.

Connector. See Message processing node connector.

Cube. Another term for a fact table. It can represent “n” dimensions, rather than just three (as may be implied by the name).

DDL (Data Definition Language)**.** A SQL statement that creates or modifies the structure of a table or database. For example, CREATE TABLE, DROP TABLE, ALTER TABLE, or CREATE DATABASE.

DML (Data Manipulation Language)**.** An INSERT, UPDATE, DELETE, or SELECT SQL statement.

Data Append. A data loading technique where new data is added to the database leaving the existing data unaltered.

Data Append. A data loading technique where new data is added to the database leaving the existing data unaltered.

Data Cleansing. A process of data manipulation and transformation to eliminate variations and inconsistencies in data content. This is typically to improve the quality, consistency, and usability of the data.

Data Federation. The process of enabling data from multiple heterogeneous data sources to appear as though it is contained in a single relational database. Can also be referred to “distributed access”.

Data mart. An implementation of a data warehouse, typically with a smaller and more tightly restricted scope, such as for a department, workgroup, or subject area. It could be independent, or derived from another data warehouse environment (dependent).

Data mart - Dependent. A data mart that is consistent with, and extracts its data from, a data warehouse.

Data mart - Independent. A data mart that is standalone, and does not conform with any other data mart or data warehouse.

Data Mining. A mode of data analysis that has a focus on the discovery of new information, such as unknown facts, data relationships, or data patterns.

Data Model. A representation of data, its definition, characteristics, and relationships.

Data Partition. A segment of a database that can be accessed and operated on independently even though it is part of a larger data structure.

Data Refresh. A data loading technique where all the data in a database is completely replaced with a new set of data.

Data silo. A stand-alone set of data in a particular department or organization used for analysis, but typically not shared with other departments or organizations in the enterprise.

Data Warehouse. A specialized data environment developed, structured, shared, and used specifically for decision support and informational (analytic) applications. It is subject oriented rather than application oriented, and is integrated, non-volatile, and time variant.

Database Instance. A specific independent implementation of a DBMS in a specific environment. For example, there might be an independent DB2 DBMS implementation on a Linux server in Boston supporting the Eastern offices, and another separate and independent DB2 DBMS on the same Linux server supporting the western offices. They would represent two instances of DB2.

Database Partition. Part of a database that consists of its own data, indexes, configuration files, and transaction logs.

DataBlades. These are program modules that provide extended capabilities for Informix® databases, and are tightly integrated with the DBMS.

DB Connect. Enables connection to several relational database systems and the transfer of data from these database systems into the SAP® Business Information Warehouse.

Debugger. A facility on the Message Flows view in the Control Center that enables message flows to be visually debugged.

Deploy. Make the configuration and topology of the broker domain operational.

Dimension. Data that further qualifies or describes a measure, such as amounts or durations.

Discovery. Data mining technique for finding patterns in data with no prior knowledge of them.

Distributed Application In message queuing, a set of application programs that can each be connected to a different queue manager, but that collectively constitute a single application.

Drill-down. Iterative analysis that explores facts at more detailed levels of the dimension hierarchies.

Dynamic SQL. SQL that is interpreted during execution of the statement.

Engine. A program that performs a core or essential function for other programs. A database engine performs database functions on behalf of the database user programs.

Enrichment. The creation of derived data. An attribute level transformation performed by some type of algorithm to create one or more new (derived) attributes.

Entity. A person, place, thing, or event of interest to the enterprise. Each entity in a data model is unique.

Extenders. These are program modules that provide extended capabilities for DB2, and are tightly integrated with DB2.

FACTS. A collection of measures, and the information to interpret those measures in a given context.

Federated data. A set of physically separate data structures that are logically linked together by some mechanism for analysis, but which remain physically in place.

Federated Server. Any DB2 server where the WebSphere Information Integrator is installed.

Federation. Providing a unified interface to diverse data.

Foreign Key. An attribute or set of attributes that refer to the primary key of another entity.

Gateway. A means to access a heterogeneous data source. It can use native access or ODBC technology.

Grain. The fundamental atomic level of data represented in a fact table. As examples, typical grains that could be used, when considering time, would be day, week, month, year, and so forth.

Granularity. The level of summarization of the data elements. It refers to the level of detail available in the data elements. The more detailed data that is available, the lower the level of granularity. Conversely, the less details that are available, the higher the level of granularity (or level of summarization of the data elements).

Instance. A particular realization of a computer process. Relative to database, the realization of a complete database environment.

Java Database Connectivity. An application programming interface that has the same characteristics as ODBC but is specifically designed for use by Java database applications.

Java Development Kit. Software package used to write, compile, debug, and run Java applets and applications.

Java Message Service. An application programming interface that provides Java language functions for handling messages.

Java Runtime Environment. A subset of the Java Development Kit that allows you to run Java applets and applications.

Key. An attribute of set of attributes that uniquely identifies an entity.

Materialized Query Table. A table where the results of a query are stored for later reuse.

Measure. A data item that measures the performance or behavior of business processes.

Message domain. The value that determines how the message is interpreted (parsed).

Message flow. A directed graph that represents the set of activities performed on a message or event as it passes through a broker. A message flow consists of a set of message processing nodes and message processing connectors.

Message parser. A program that interprets the bit stream of an incoming message and creates an internal representation of the message in a tree structure. A parser is also responsible for generating a bit stream for an outgoing message from the internal representation.

Meta Data. Typically called data (or information) about data. It describes or defines data elements.

MOLAP. Multidimensional OLAP. Can be called MD-OLAP. It is OLAP that uses a multidimensional database as the underlying data structure.

Multidimensional analysis. Analysis of data along several dimensions. For example, analyzing revenue by product, store, and date.

Multi-Tasking. Operating system capability that allows multiple tasks to run concurrently, taking turns using the resources of the computer.

Multi-Threading. Operating system capability that enables multiple concurrent users to use the same program. This saves the overhead of initiating the program multiple times.

Nickname. An identifier that is used to reference the object located at the data source that you want to access.

Node Group. Group of one or more database partitions.

Node. See Message processing node and Plug-in node.

Non-Additive Measure. Measure of a fact that cannot be added across any of its dimensions, such as a percentage.

ODS. (1) Operational data store: A relational table for holding clean data to load into InfoCubes, and can support some query activity. (2) Online Dynamic Server - an older name for IDS.

OLAP. OnLine Analytical Processing. Multidimensional data analysis, performed in real time. Not dependent on underlying data schema.

Open Database Connectivity. A standard application programming interface for accessing data in both relational and non-relational database management systems. Using this API, database applications can access data stored in database management systems on a variety of computers even if each database management system uses a different data storage format and programming interface. ODBC is based on the call level interface (CLI) specification of the X/Open SQL Access Group.

Optimization. The capability to enable a process to execute and perform in such a way as to maximize performance, minimize resource utilization, and minimize the process execution response time delivered to the user.

Partition. Part of a database that consists of its own data, indexes, configuration files, and transaction logs.

Pass-through. The act of passing the SQL for an operation directly to the data source without being changed by the federation server.

Pivoting. Analysis operation where user takes a different viewpoint of the results. For example, by changing the way the dimensions are arranged.

Primary Key. Field in a table record that is uniquely different for each record in the table.

Predictive Methods. Data mining techniques, such as classification and regression, to predict categorical or numeric values.

Process. An instance of a program running in a computer.

Program. A specific set of ordered operations for a computer to perform.

Pushdown. The act of optimizing a data operation by pushing the SQL down to the lowest point in the federated architecture where that operation can be executed. More simply, a pushdown operation is one that is executed at a remote server.

Relationship. The business rule that associates entities.

ROLAP. Relational OLAP. Multidimensional analysis using a multidimensional view of relational data. A relational database is used as the underlying data structure.

Roll-up. Iterative analysis, exploring facts at a higher level of summarization.

Scoring. Data mining method used against the data to create a data mining model.

Semi-additive Measure. Measure of a fact that can be added across only some of its dimensions, such as a balance.

Server. A computer program that provides services to other computer programs (and their users) in the same or other computers. However, the computer that a server program runs in is also frequently referred to as a server.

Shared nothing. A data management architecture where nothing is shared between processes. Each process has its own processor, memory, and disk space.

Spreadmart. A stand-alone, non-conforming, non-integrated set of data, such as a spreadsheet, used for analysis by a particular person, department, or organization.

Static SQL. SQL that has been compiled prior to execution. Typically provides the best performance.

Static SQL. SQL that has been compiled prior to execution. Typically provides the best performance.

Subject Area. A logical grouping of data by categories, such as customers or items.

Synchronous Messaging. A method of communication between programs in which a program places a message on a message queue and then waits for a reply before resuming its own processing.

Task. The basic unit of programming that an operating system controls. Also see Multi-Tasking.

Thread. The placeholder information associated with a single use of a program that can handle multiple concurrent users. Also see Multi-Threading.

Type Mapping. The mapping of a specific data source type to a DB2 UDB data type.

Unit of Work. A recoverable sequence of operations performed by an application between two points of consistency.

User Mapping. An association made between the federated server user ID and password and the data source (to be accessed) user ID and password.

Virtual Database. A federation of multiple heterogeneous relational databases.

Warehouse Catalog. A subsystem that stores and manages all the system meta data.

Wrapper. The means by which a data federation engine interacts with heterogeneous sources of data. Wrappers take the SQL that the federation engine uses and maps it to the API of the data source to be accessed. For example, they take DB2 SQL and transform it to the language understood by the data source to be accessed.

xtree. A query-tree tool that allows you to monitor the query plan execution of individual queries in a graphical environment.

Abbreviations and acronyms

ACS	Access Control System.
ADK	Archive Development Kit.
AIX®	Advanced Interactive eXecutive from IBM.
API	Application Programming Interface.
AQR	Automatic Query Re-write.
AR	Access Register.
ARM	Automatic Restart Manager.
ART	Access Register Translation.
ASCII	American Standard Code for Information Interchange.
AST	Application Summary Table.
BLOB	Binary Large OBject.
BW	Business Information Warehouse (SAP).
CCMS	Computing Center Management System.
CFG	Configuration.
CLI	Call Level Interface.
CLOB	Character Large OBject.
CLP	Command-Line Processor.
CORBA	Common Object Request Broker Architecture.
CPU	Central Processing Unit.
CS	Cursor Stability.
DAS	DB2 Administration Server.
DB	Database.
DB2	Database 2™.
DB2 UDB	DB2 Universal Database.
DB2W	DB2 Warehouse.
DBA	Database Administrator.
DBM	Database Manager.
DBMS	Database Management System.
DCE	Distributed Computing Environment.
DCM	Dynamic Coserver Management.
DCOM	Distributed Component Object Model.
DDL	Data Definition Language. a SQL statement that creates or modifies the structure of a table or database. For example, CREATE TABLE, DROP TABLE.
DES	Data Encryption Standard.
DIMID	Dimension Identifier.
DLL	Dynamically Linked Library.
DMDL	Dimensional Modeling Design Life Cycle.
DML	Data Manipulation Language. An INSERT, UPDATE, DELETE, or SELECT SQL statement.
DMS	Database Managed Space.
DPF	Data Partitioning Facility.
DRDA®	Distributed Relational Database Architecture™.
DSA	Dynamic Scalable Architecture.
DSN	Data Source Name.
DSS	Decision Support System.
DWE	Data Warehouse Edition.
EAI	Enterprise Application Integration.
EAR	Entity, Attribute, Relationship data model. Also denoted by E/R.

EBCDIC	Extended Binary Coded Decimal Interchange Code.
EDA	Enterprise Data Architecture.
EDU	Engine Dispatchable Unit.
EDW	Enterprise Data Warehouse.
EGM	Enterprise Gateway Manager.
EJB™	Enterprise Java Beans.
ELT	Extract-Load-Transform.
E/R	Enterprise Replication.
E/R	Entity, Attribute, Relationship data model. Also denoted by EAR.
ERP	Enterprise Resource Planning.
ESE	Enterprise Server Edition.
ETL	Extract, Transform, and Load.
ETTL	Extract, Transform/Transport, and Load.
FJS	Filter, then Join, then Sort.
FP	Fix Pack.
FTP	File Transfer Protocol.
Gb	Gigabits.
GB	Gigabytes.
GUI	Graphical User Interface.
HADR	High Availability Disaster Recovery.
HDR	High availability Data Replication.
HPL	High Performance Loader.
I/O	Input/Output.
IBM	International Business Machines Corporation.
ID	Identifier.
IDE	Integrated Development Environment.
IDS	Informix Dynamic Server.
II	Information Integrator.
IMG	Integrated Implementation Guide (for SAP).
IMS™	Information Management System.
ISAM	Indexed Sequential Access Method.
ISM	Informix Storage Manager.
ISV	Independent Software Vendor.
IT	Information Technology.
ITR	Internal Throughput Rate.
ITSO	International Technical Support Organization.
IX	Index.
J2EE	Java 2 Platform Enterprise Edition.
JAR	Java Archive.
JDBC	Java Database Connectivity.
JDK™	Java Development Kit.
JE	Java Edition.
JMS	Java Message Service.
JRE™	Java Runtime Environment.
JVM™	Java Virtual Machine.
KB	Kilobyte (1024 bytes).
LDAP	Lightweight Directory Access Protocol.
LPAR	Logical Partition.
LV	Logical Volume.
Mb	Megabits.
MB	Megabytes.
MDC	Multidimensional Clustering.
MPP	Massively Parallel Processing.
MQI	Message Queuing Interface.
MQT	Materialized Query Table.
MRM	Message Repository Manager.

MTK	DB2 Migration ToolKit for Informix.
NPI	Non-Partitioning Index.
ODBC	Open Database Connectivity.
ODS	Operational Data Store.
OLAP	OnLine Analytical Processing.
OLE	Object Linking and Embedding.
OLTP	OnLine Transaction Processing.
ORDBMS	Object Relational Database Management System.
OS	Operating System.
O/S	Operating System.
PDS	Partitioned Data Set.
PIB	Parallel Index Build.
PMML	Predictive Modeling Markup Language.
PSA	Persistent Staging Area.
RBA	Relative Byte Address.
RBW	Red Brick™ Warehouse.
RDBMS	Relational Database Management System.
RID	Record Identifier.
RR	Repeatable Read.
RS	Read Stability.
SCB	Session Control Block.
SDK	Software Developers Kit.
SDLC	Software Development Life Cycle.
SID	Surrogate Identifier.
SMIT	Systems Management Interface Tool.
SMP	Symmetric MultiProcessing.
SMS	System Managed Space.
SOA	Service Oriented Architecture.
SOAP	Simple Object Access Protocol.
SPL	Stored Procedure Language.
SQL	Structured Query.
SQW	In DB2 Warehouse, it is the SQL warehousing tool.
TCB	Thread Control Block.
TMU	Table Management Utility.
TS	Tablespace.
UDB	Universal Database.
UDF	User Defined Function.
UDR	User Defined Routine.
URL	Uniform Resource Locator.
VG	Volume Group (RAID disk terminology).
VLDB	Very Large Database.
VP	Virtual Processor.
VSAM	Virtual Sequential Access Method.
VTI	Virtual Table Interface.
WSDL	Web Services Definition Language.
WWW	World Wide Web.
XBSA	X-Open Backup and Restore APIs.
XML	eXtensible Markup Language.
XPS	Informix eXtended Parallel Server.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 522. Note that some of the documents referenced here may be available in softcopy only.

- *Data Mart Consolidation: Getting Control of Your Enterprise Information*, SG24-6653
- *DB2 Cube Views: A Primer*, SG24-7002
- *DB2 UDB's High-Function Business Intelligence in e-business*, SG24-6546
- *Dimensional Modeling: In a Business Intelligence Environment*, SG24-7138
- *Enhance Your Business Applications: Simple Integration of Advanced Data Mining Functions*, SG24-6879
- *Leveraging DB2 Data Warehouse Edition for Business Intelligence*, SG24-7274
- *Mining Your Own Business in Health Care Using DB2 Intelligent Miner for Data*, SG24-6274
- *Preparing for DB2 Near-Realtime Business Intelligence*, SG24-6071
- *Up and Running with DB2 UDB ESE Partitioning for Performance in an e-Business Intelligence World*, REDP-6917

Other publications

These publications are also relevant as further information sources:

- “Accessible Insight: Embedded Data Mining” by J.B. Rollins and R. Hale in DB2 Magazine, Quarter 4, 2005
- Anderson, *An Introduction to Neural Networks*, MIT Press, 1995, ISBN 0262011441
- Apostolos-Paul, (Editor): *Neural Networks in the Capital Markets*, Wiley, 1995, ISBN 0471943649

- *DB2 Administration Guide: Performance*, SC10-4222
- *DB2 Alphablox Cube Server Administrator's Guide*, SC18-9433
- DeWilde, *Neural Network Models, 2nd Edition*, Springer, 1997 ISBN 3540761292
- Efroymson, “Multiple regression analysis", in Ralston and Wilf (ed.), *Mathematical Methods for Digital Computers*, Wiley, 1960, ISBN 0471706892
- “Embedded Data Mining: Steps to Success” by R. Hale and J.B. Rollins in Computerworld, March 13, 2006
- Fox, *Applied Regression Analysis, Linear Models and Related Methods*, Sage, 1997, ISBN 080394540X
- Hale, C.R. and Rollins, J.B., “Process and Heuristic Statistic for Prospect Selection through Data Mining”, SVL920030124US1 patent pending (Dec. 2003)
- *IBM DB2 Data Warehouse Edition Administration and Programming for Intelligent Miner Modeling, Version 9.1,* SH12-6838
- *IBM DWE Miningblox: Administration and Programming Guide*, SC18-9805
- *Informix JDBC Driver - Programmer’s Guide V1.4*, Part No. 000-5343
- Jain, et al, *Data Clustering: A Review*, ACM Comp. Surv., 31, 264-323
- Kmenta, *Elements of Econometrics, 2nd Ed.*, University of Michigan Press, 1997, ISBN 0472084763
- “Knowledge Discovery in Databases: An Overview” by W. Frawley, et al, in AI Magazine, Fall 1992
- Kraemer, et al, *System and Method to Optimize Control Cohorts Using Clustering Algorithms*, END920060045US1 patent pending (Oct. 2006)
- “Neural Networks and Statistical Models” by W. Sarle in Proceedings of the Nineteenth Annual SAS Users Group International Conference, April 1994
- Pyle, *Data Preparation for Data Mining*, Morgan Kaufmann Publishers, 1999, ISBN 1558605290
- “Techniques of Cluster Algorithms in Data Mining” by R. Grabmeier and A. Rudolph in Data Mining and Knowledge Discovery, Vol. 6, No. 4, 2002

Online resources

These Web sites are also relevant as further information sources:

- CRoss Industry Standard Process for Data Mining (CRISP-DM)

  http://www.crisp-dm.org/

- The Data Mining Group

  http://www.dmg.org/
  http://www.dmg.org/pmml-v3-0.html

- DB2 Basics: An introduction to materialized query tables

  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0509melnyk/

- DB2 Cube Views Version 8.2 Best Practices

  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0511pay/

- *Embedded Analytics in IBM DB2 Universal Database for Information on Demand*, Information Integration Software Solutions white paper (Aug. 2003), found at:

  ftp://ftp.software.ibm.com/software/data/pubs/papers/embeddedanalytics.pdf

- IBM developersWork Web site: DB2 Data Warehouse OLAP Services, Part 1: Starting out with OLAP services

  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0606gong/index.html

- Integrate DB2 Alphablox and DB2 Cube Views to build multidimensional OLAP Web apps

  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0602wen/index.html#download

- Pearson's correlation coefficient

  http://www.stat.tamu.edu/stat30x/notes/node39.html

- Referential Integrity Utility for IBM DB2 Cube Views

  http://www.alphaworks.ibm.com/tech/riu4db2cv

- "Transform Regression and the Kolmogorov Superposition Theorem" by E. Pednault in Proceedings of the 2006 SIAM Conference on Data Mining, April 2006

  http://www.siam.org/meetings/sdm06/proceedings.htm

- "Unsupervised Learning" by Z. Ghahramani in O. Bousquet, G. Raetsch, and U. von Luxburg, eds, *Advanced Lectures on Machine Learning*, Vol. 3176 of Lecture Notes in Artificial Intelligence, Springer Verlag, Heidelberg, Germany, found at:

  http://www.gatsby.ucl.ac.uk/~zoubin/course05/ul.pdf

How to get IBM Redbooks

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy IBM Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

C

D

E

F

G

H

N

O

P

Q

R

S

T

U

V

W

X

Dynamic Warehousing: Data Mining Made Easy

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Dynamic Warehousing: Data Mining Made Easy

Redbooks

Mining information from your data warehouse environment

Understanding the data mining opportunities

Using DB2 Warehouse mining capabilities

Data mining has evolved from the ethereal domain of the highly-skilled mathematician to the expert data miner's workbench tool and ultimately to widely accessible business applications. For decades, industry and academia have been engaged in far-reaching research and development of data mining. At the same time, businesses have been leveraging this research, exploiting a handful of algorithms most useful in finding information to help resolve business problems.

Recent trends have made these algorithms and systems, which are rooted in solid research, available to a wide range of business users in easy-to-use forms. Large numbers of business analysts, who may not be data mining experts, can now solve high-value business problems using data mining technology embedded in database-resident business applications.

In this IBM Redbooks publication, we discuss the methodology and selected techniques of embedded data mining and show how sophisticated technologies can be used in today's business environment to create significant business value. All this is enabled by the IBM DB2 Warehouse (DB2W) data mining capabilities. Using DB2W, we show examples of using data mining capabilities for such analytic functions as data modeling, scoring, and visualization. In addition, there are scenarios and examples that help in understanding where, when, and how to use data mining. For techniques, technical details, and practical examples, this is the book you need.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7418-00 ISBN 0738488860