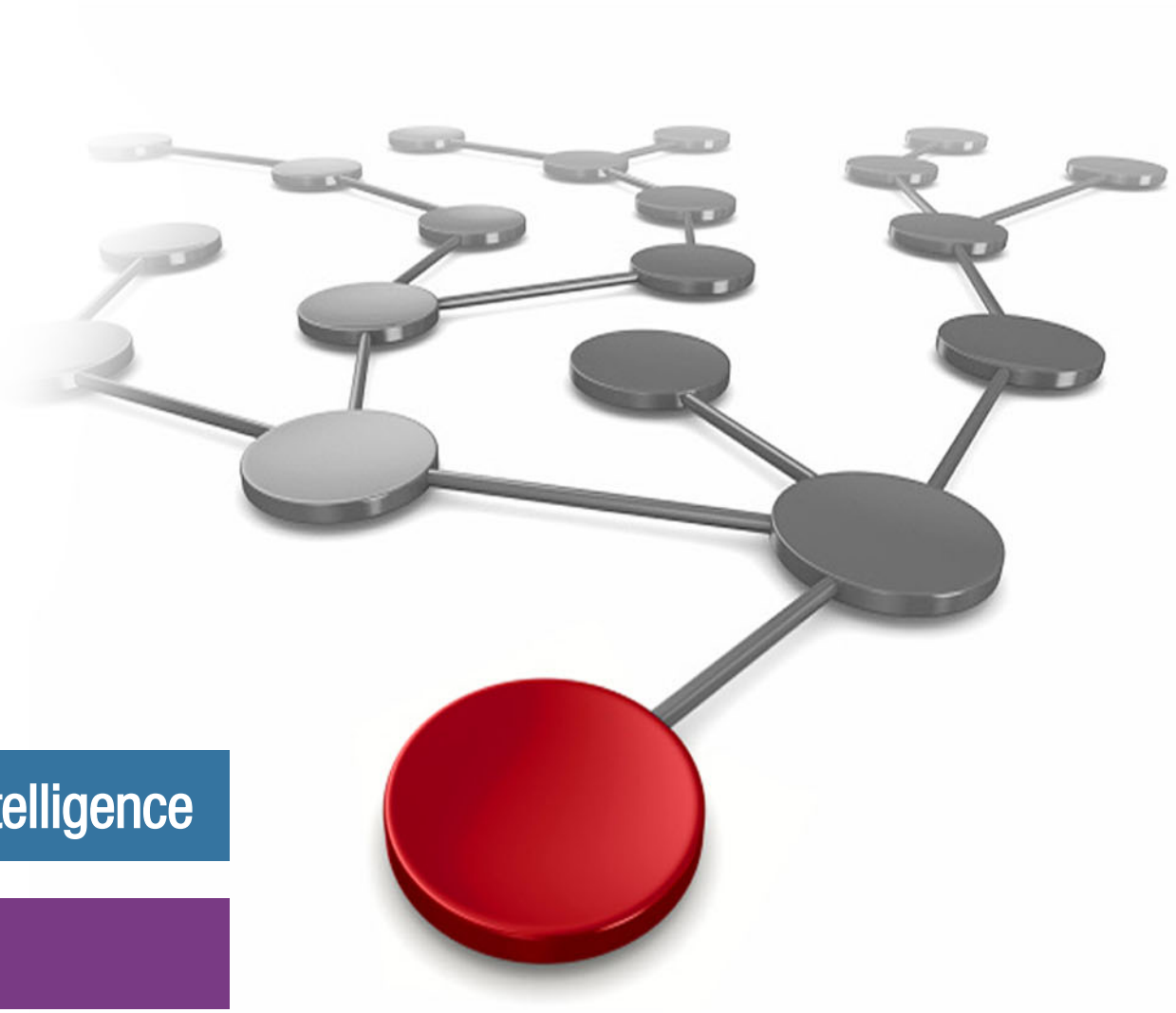
Draft Document for Review January 26, 2024 4:43 pm SG24-8552-00

Redbooks
ibm.com/redbooks

Turning Data into Insight with Machine Learning for IBM z/OS

Abid Alam
Roger Bales
Vineet Dumir
Nicholas Kunze
Jia Li
Sunil Mishra
Evan Rivera
Meng Wan
Yichong Yu



Artificial Intelligence

IBM Z

IBM

Redbooks

IBM Redbooks

Turning Data into Insight with Machine Learning for IBM z/OS

January 2024

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

First Edition (January 2024)

This edition applies to Machine Learning for IBM z/OS version 3.1.

This document was created or updated on January 9, 2024.

Contents

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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 actual people or business enterprises 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at https://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

CICS®
Cloudant®
DataStage®
DB2®
Db2®
GDPS®
IBM®
IBM Cloud®
IBM Cloud for Financial Services®
IBM Cloud Pak®
IBM Cloud Satellite®
IBM Research®
IBM Security®
IBM Telum®
IBM Watson®
IBM Z®
IBM z13®
IBM z16™
InfoSphere®
Promontory®
RACF®
Redbooks®
Redbooks (logo) ®
Satellite™
The AI Ladder®
WebSphere®
z Systems®
z/OS®
z13®
z15®
z16™

The following terms are trademarks of other companies:

Evolution, are trademarks or registered trademarks of Kenexa, an IBM Company.

Intel, Intel Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware, and the VMware logo are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

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

Preface

The exponential growth in data over the last decade coupled with a drastic adoption of artificial intelligence has opened organizations up to valuable insights, opportunities for improvements in efficiency, increases in profits, and overall deliver more business value both internally to employees and shareholders and externally to clients and consumers. The data that fuels these insights is the foundation that these organizations must tap in to innovate and they must capitalize on the most advanced AI capabilities to stay ahead of the competition, and they must do so in a secure environment that protects the data throughout its lifecycle and data access in real-time at any time.

When it comes to security, nothing can rival IBM® Z, the multi-workload transactional platform that powers the core business processes of the majority of the Fortune 500 enterprises with unmatched security, availability, reliability, and scalability. With core transactions and data originating on IBM Z®, it simply makes sense for AI to exist and run on the same platform.

For years, some businesses chose to move their sensitive data off IBM Z, however, the massive growth of digital data, the punishing cost of security exposures as well as the unprecedented demand for instant actionable intelligence from data in real time have convinced them to rethink that decision and, instead, embrace the strategy of data gravity for AI. Transaction gravity is the other force here which eludes to the trillions of transactions that move through IBM Z as it hosts some of the worlds most mission critical workloads that runs the global economy. It is this dual strategy of data and transaction gravity that sets IBM Z apart. For organizations to get real-time business and operational insights at scale, the AI workloads must be co-located with the transactional applications and where the data originates.

IBM Z has made tremendous invests in this space, starting at the hardware silicon level with the industry first Telum on-chip AI inference accelerator introduced with IBM z16™ all the way up the stack with our enterprise grade AI/ML software offerings with a robust and optimized open source ecosystem support.

Machine Learning for IBM z/OS® is one of IBM's flagship offerings that helps organizations deploy machine learning models on IBM z/OS environments natively close to their most mission critical workloads and infuse them with AI. In the inherently secure Z environment, your machine learning scoring services can co-exist with your transactional applications and data, supporting high throughput and minimizing response time while delivering consistent service level agreements (SLAs).

This book introduces Machine Learning for IBM z/OS version 3.1.0 and describes its unique value proposition in addition to the business value of AI on the IBM Z platform. It provides guidance for you to get started with aligning your business goals and use cases for AI. It outlines the various patterns for accessing and consuming IBM Z data in on-premises, cloud, and hybrid environments. It discusses in detail the steps and best practices for model building, deployment, and inferencing and post deployment monitoring.

The book includes examples of how you can use the versatile and intuitive web user interface of MLz to quickly train, evaluate, and deploy a model. Through a batch job elapsed time prediction example, the book explores the value of MLz in such a use case and shows how to apply the batch job elapsed time model with MLz for application integration and provides insight into how MLz can be leveraged within machine learning operations to operationalize AI on IBM Z. Most importantly, it examines use cases across industries to illustrate how you can easily turn your massive data into valuable insights with Machine Learning for IBM z/OS.

Authors

This book was produced by a team of specialists from around the world.

Abid Alam is a Sr. Product Manager in the AI for IBM Z team. He has over 4 years of industry experience in product management strategy and operations. He is responsible for driving the adoption of AI on IBM Z and works closely with clients and IBM product development and design teams. Prior to this role, Abid has worked in the startup and consulting space with a focus on go-to-market and client adoption strategy and driving industry partnerships.

Roger Bales is Program Director – IBM z/OS Strategy. He has over 40 years of IBM Z mainframe systems and software experience. He is responsible for z/OS strategy and works closely with IBM Z clients and across the IBM product management, development, and sales teams. Previously, he led the global IBM Z Consulting Services Practice. Prior to IBM, Mr. Bales held various technical and management positions at AT&T and served as General Manager for a global ISV.

Vineet Dumir is an AI Solution Architect in IBM India Systems Lab. He has 14 years of experience in the database administration, systems programming and machine learning. His areas of expertise include artificial intelligence, machine learning, data science, data engineering, IBM Db2® database administration and Z systems programming.

Nicholas Kunze is a Client Engineer with the Client Engineering for Systems team. He has 7 years of experience in infrastructure and development, and 5 years of experience on IBM Z with a focus on performance testing and benchmarking for data engineering and AI/ML.

Jia Li is a Software Engineer in IBM Silicon Valley Lab with 20 years of experience in software design and development. She has expertise in database administration, database application development and machine learning tools development.

Sunil Mishra is a Senior Software Engineer in the United States. He has over 25 years of experience in the software development, design, and architecture field. He has worked at IBM for 17 years and his areas of expertise include enterprise application development and design for health care, government, retail, and so on.

Evan Rivera is on the AI development and product management team on IBM Z in the IBM Silicon Valley Lab. He has 5 years of experience with development, design, architecture, and product management within the AI and ML field. He is responsible for bringing AI products to the market and driving client adoption of AI on IBM Z.

Meng Wan is a Senior Software Engineer in China. He has 13 years of experience in Z system programming, software development and design for machine learning on IBM z/OS. He has worked at IBM for 13 years and his areas of expertise include artificial intelligence, machine learning, data science, and Z programming.

Yichong Yu is a Senior Solutions Architect in IBM Cloud® for Financial Services. She has over 20 years of software development, system design, and research experience, and over 15 years of experience in leading software engineering teams that deliver cutting edge systems in an agile environment. She has delivered solutions for different industries, including financial, manufacturing, food, retail, health care, telecommunication, and transportation.

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

Makenzie Manna
IBM Redbooks®, Poughkeepsie Center

Andrew Sica, Steve Warren, Kelly Yang, Maggie Lin, Maryela Weihrauch, Sueli Almeida, Paul Cadarette, Greg Vance, Elpida Tzortzatos, Krishna Ratakonda
IBM USA

Sharon Yu, Deng Ke Zhao
IBM China

Daniel Martin, Khadija Souissi, Markus Wolff, Jan Hofmeier
IBM Germany

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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 books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on LinkedIn:

https://www.linkedin.com/groups/2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/subscribe

- Stay current on recent Redbooks publications with RSS Feeds:

  https://www.redbooks.ibm.com/rss.html

1

Unlocking the business value of AI on IBM Z

Artificial intelligence (AI) adoption has helped companies become more efficient, increase profits and ultimately deliver more business value to their clients. We are rapidly moving from a world where AI was seen as a nice-to-have option for the business strategy, to a place now where AI is essential to business, one might position it as, “AI first”.

In a recent survey of over 3,000 CEOs from companies around the world, conducted by the IBM Institute for Business Value found that 75% believed that competitive advantage will be driven in part by who has the most advanced AI capabilities.[1]

In this chapter, we will dive into the discussion around the business value of AI, and in particular the immense value achieved and opportunities unlocked from implementing AI within your IBM Z environment.

[1] https://www.ibm.com/thought-leadership/institute-business-value/c-suite-study/ceo

1.1 AI trust and transparency: IBM’s commitment

For decades, IBM has followed a set of core principles that have helped guide the handling of client data and insights in addition to enabling responsible development and deployment of new, transformative technologies and innovations. It is essential that all technology companies adopt similar principles, as we see governments around the world increasingly regulating AI, for example, the EU AI Act and the United States Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence.

The IBM trust and transparency principles include:

- The purpose of AI is to augment human intelligence - AI should make all of us better at our jobs, the benefits of the AI era should touch the many, not just the elite few.
- Data and insights belong to their creator, IBM clients - Data is their data, and their insights are their insights. Government data policies should be fair, equitable and prioritize openness.
- Technology must be transparent and explainable - Companies must be clear about who trains their AI systems, what data was used in training and, most importantly, what went into their algorithms' recommendations.

For more details on IBM’s Principles for Trust and Transparency, see:

https://www.ibm.com/policy/trust-transparency-new/

Today, AI can be generally classified into the following three areas:

- Reactive machines

  Reactive machines are the most basic types of AI systems. They are task specific with no memory-based functionality, thus they lack the ability to learn from past experiences and will always react to specific inputs with predefined outputs.

  Some examples of reactive machines include the following:

  – IBM Deep Blue was a chess-playing expert system who famously planed world chess champion Garry Kasparov in 1996 winning two games and losing for. In 1997, after further refinements Deep Blue won two games and played to a draw on three games. This is a good example of AI reacting to external data to augment decision making and performance. Deep Blue's victory is considered a milestone in the history of artificial intelligence.

  – In February 2011 IBM's Watson computer competed on Jeopardy! against the TV quiz shows two biggest all-time champions. Watson is a computer running software called Deep QA, developed by IBM Research®. While the grand challenge driving the project was to win on Jeopardy!, the broader goal of Watson was to create a new generation of technology that can find answers in unstructured data more effectively than standard search technology.

  Both the IBM Deep Blue and Jeopardy! experiences represent examples of advancing AI technologies designed to react and augment decision-making.

- Machine learning and Deep Learning

  Machine learning (ML) is a subset of artificial intelligence that focuses on the development of algorithms and models allowing systems to learn and improve from experience without explicit programming. It encompasses various techniques such as supervised, unsupervised, and reinforcement learning. Supervised learning involves training models on labeled data to make predictions or classifications. Unsupervised learning deals with discovering patterns or structures within unlabeled data. Reinforcement learning revolves

around training agents to make decisions by rewarding desired behaviors. Machine learning is currently used widely in applications across many different industries. In finance and banking, it aids use cases like fraud detection, anti-money laundering and algorithmic trading. In health care, it assists in diagnostics and personalized medicine. Moreover, recommendation systems in e-commerce and advertisement also significantly leverages machine learning.

Deep learning (DL), a subset of machine learning, utilizes neural networks with multiple layers to extract hierarchical representations from data. These networks, inspired by the human brain's structure, consist of interconnected nodes that process information. Deep learning models, such as convolutional neural networks (CNNs) for image recognition or recurrent neural networks (RNNs) for sequential data, have revolutionized various fields. In computer vision, deep learning powers object detection, image classification, and facial recognition systems. Natural language processing (NLP) heavily relies on deep learning for tasks like machine translation, sentiment analysis, and chatbots. Additionally, autonomous vehicles leverage deep learning for perception and decision-making, marking its significant impact on modern technology.

- Generative AI and its applications

  Generative AI encompasses models that create new content resembling real data, often based on patterns and structures learned from existing datasets. This category includes generative adversarial networks (GANs), variational autoencoders (VAEs), and transformers. GANs, for instance, consist of two neural networks, a generator, and a discriminator, competing to produce authentic-looking data. Applications of generative AI are diverse. In art and media, it aids in generating realistic images, music, and text. Furthermore, it contributes to data augmentation techniques for improving training datasets in machine learning. In drug discovery, generative AI assists in molecule generation and drug design by generating novel compounds with desired properties. Additionally, it finds use in creating synthetic data for training models while preserving privacy and security in sensitive domains.

1.1.1 Using AI to create a competitive advantage

The rate and pace of digital transformation has accelerated and with that transformation the adoption and use of AI has grown as well. AI is becoming an economic accelerator for companies today and its business impact is positively affecting both top line revenue through enhanced customer engagement and support, at the same time it is improving the bottom line by reducing operational costs driven by increased employee and resource productivity. AI is now impacting both the top and bottom lines for many businesses and, as such, is seen as an essential initiative for future competitiveness.

The data backs this up, in a recent global survey of over 6000 C-level executives, companies reported 6.3% points of direct revenue growth attributable to AI. At the same time more than 85% of advanced AI adopters are reducing operating costs with AI. Out of the 85%, 47% of them have been able to reach cost improvements through process efficiencies, 41% in supply chain and production and 39% in human resource efficiency improvements.[2]

Chapter 2, “Building business-driven and strategic use cases with MLz” on page 15 provides a comprehensive view of AI use cases and goes deeper into key use case development techniques including planning, ideation, assessment, and execution.

[2] IBM Institute for Business Value Study, “The Business Value of AI”, November 2020, https://www.ibm.com/thought-leadership/institute-business-value/en-us/report/ai-value-pandemic

1.2 AI methodology overview

As AI continues to evolve and grow, so do the capabilities and applicability of AI across many different domains and tasks, as shown in Figure 1-2.

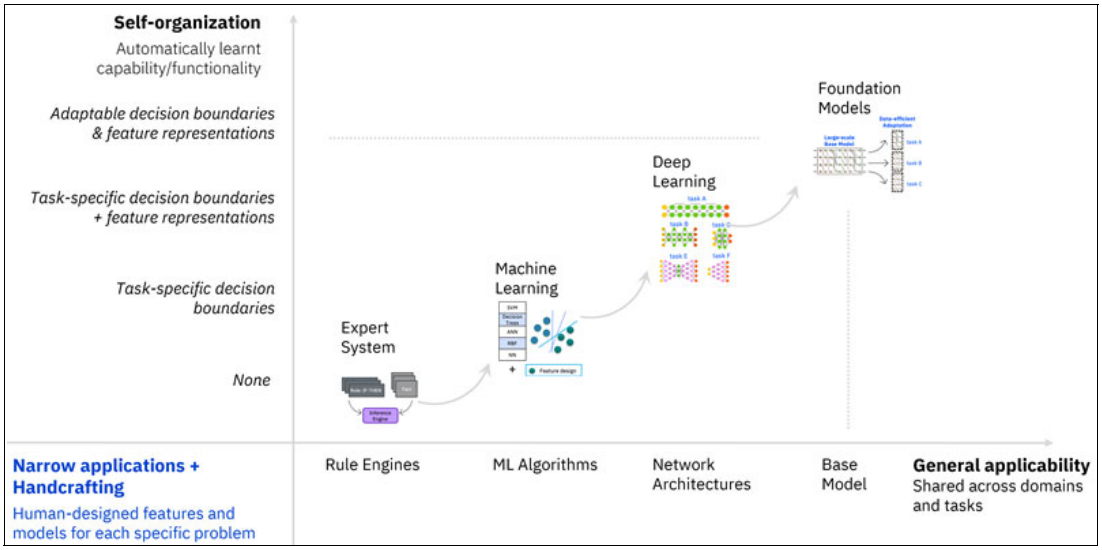


Figure 1-1 Evolution® of AI

Although AI is now mainstream, it is important to know that the initial concepts have been around since the mid-1900s. Over the years, the evolution of AI and supporting technologies that provide the necessary computational resources have enabled enterprises to leverage it within the real world.

Prior to leveraging AI, enterprises heavily utilized rule-based systems, and still do. Rule based systems apply logical rules generated by a human to perform data handling and manipulation. These rules have already been predefined.

With the emergence of AI, enterprises can enhance or replace rule-based systems. Following John McCarthy's definition of AI, AI “is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable.”[3]

AI as an overall umbrella is very widespread. Taking a deeper dive, machine learning and deep learning (DL) have gained tremendous adoption within the industry. They are both considered subsets of AI, while DL is more specifically a subset of machine learning while using neural networks for picture or pattern recognition.

Generative AI is the latest iteration of innovation within the AI industry. Generative AI refers to deep-learning models that can take raw data - say, all of Wikipedia or the collected works of Rembrandt - and “learn” to generate statistically probable outputs when prompted.[4]

Additionally, foundation models are a part of generative AI that is trained on a significant amount of unlabeled data and fine-tuned for different types of use cases. The reduced number of requirements on data handling for generative AI has driven AI into the mainstream and is allowing for an acceleration of adoption in various industries. Although generative AI brings exciting new opportunities to enterprises, it normally requires more computation

[3] What is Artificial Intelligence? By John McCarthy - https://www-formal.stanford.edu/jmc/whatisai.pdf
[4] What is artificial intelligence (AI)? - https://www.ibm.com/topics/artificial-intelligence

powers to train and run inference, therefore the existing ML and DL models may still make the most sense to enterprises.

1.3 AI in today's enterprises: opportunities and challenges

Opportunities to use AI to drive business results align in two broad areas, revenue growth and productivity and cost savings. The common thread across many of the revenue growth opportunities are related to how AI can enhance the value of a good or service delivered, as well as the expansive possibilities of new business models introduced with AI. The ability to leverage data, existing knowledge, and prediction are key ingredients to enhancing product and client capabilities and value. Cost savings opportunities span a growing spectrum of areas including supply chain and inventory management as well as human resource productivity.

Headwinds for AI adoption are most often based on either cost to build, deploy and operate new AI capabilities or resistance to adopt change for several reasons, not least of which is the fear that AI will subsume the expertise and value that humans bring to a given business process or function today.

1.4 AI operational patterns: sampling vs. comprehensive

There are several AI deployment methods that vary based on use case characteristics and the functional requirements of the AI capability desired. In general, the deployment methods can be broken down in to two categories: Data sampling and Comprehensive data analysis.

Data sampling

Data Sampling, where a sample of data is all that is needed to enable the desired AI functionality and result. AI applications that act upon visual data such as images and video are some examples where sampling is used.

Comprehensive data analysis

Comprehensive Data Analysis is the second major category. This method is preferred when all data should be considered. AI functionality that needs to act on all data or transactions utilize this method. Credit Card approval or fraud detection are some examples where it is both desirable and necessary to operate on every transaction in real time. Machine Learning for IBM z/OS leveraging the scalability of the IBM Z compute platform are designed with this comprehensive, real-time, in transaction capability in mind and are enabling clients to achieve 100% of their transactions with real-time AI capabilities that sampling methods and solutions are challenged to deliver on.

1.5 AI model lifecycle overview

Atypical AI project consists of a set of steps that build an AI model lifecycle. As outlined in Figure 1-2 on page 6, each step in the lifecycle builds on the previous steps by using outputs and decision interlocks from preceding steps to form the basis of successful execution in each subsequent step.

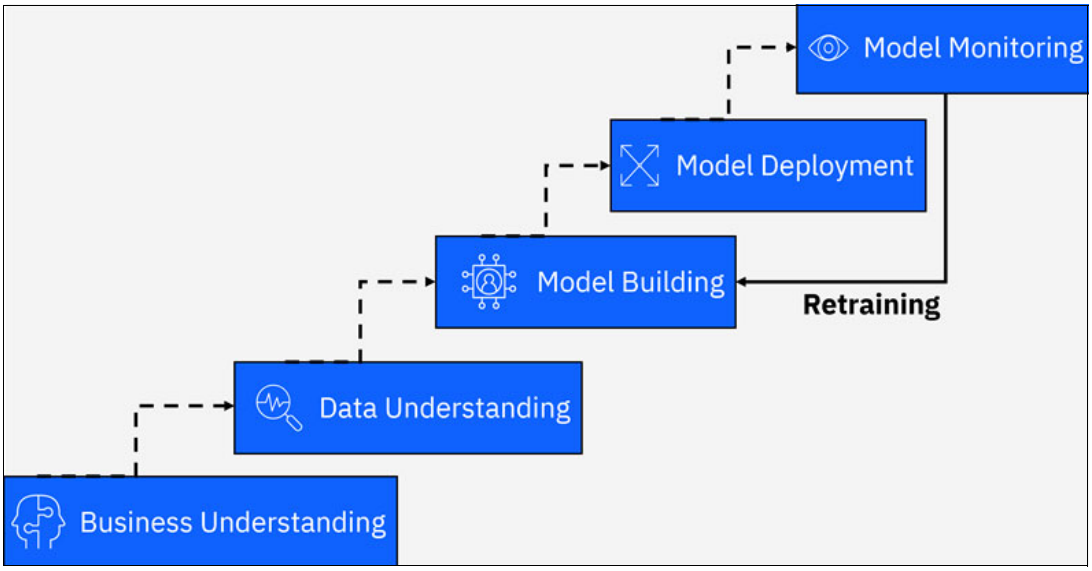


Figure 1-2 AI model lifecycle

The following sections take a more detailed look at each of the AI model lifecycle phases.

Business understanding

As use cases grow rapidly for AI, the business understanding phase become increasingly important to overall success of the AI project. In this phase of the AI model lifecycle clear and specific business goals are defined and approved by management. Goals should be described in business terms and describe quantifiable outcomes. Two elements which should be defined at a high level are the “what” and the “how”.

An example of the “what”, could be, we want to reduce high risk credit authorizations by 5% in the next 18 months. This specific goal helps keep the subsequent project activities aligned with the business goal and desired value.

When defining the “how”, the goal is to capture a high-level view of how the business goal will be accomplished. An example of the “how”, related to the sample business goal, could be, we will evaluate each request for credit, combing actual payment history and promptness, with recent buying history and existing credit score to predict future propensity to pay in full and within the repayment window. This “how” provides enough information to guide the next steps in the AI project without getting too deeply entangled in detail prematurely.

The final critical activity in this step involves clear, comprehensive communications across the business and technical leadership team obtaining approvals from each function before proceeding. A best practice is to reduce the outputs of this phase to writing so that approvals can be confirmed in writing as well as retaining the agreed-to outputs for future reference.

Data understanding

Identifying and understanding data needed to successfully build and deploy the AI project is critical to success and optimizing project productivity. Using the Business Understanding phase outputs, project specialists need to breakdown the goals including the “what” and “how”, identifying what data is required to achieve the business and technical goals. A recommended practice is to start with the desired outputs and ask the question what data is needed to produce the results as stated in the project goals. It is possible that there may be intermediate data required so it may be advisable to ask the question repeatedly, for each collection of data identified. When the data needed is identified the next step is to determine

what data exists and where and what data may need to be produced or calculated. Creation of a data map is useful in helping to understand data needed and to build effective plans and methods to acquire or build the data.

Iterative data insights development

Chapter 3, “The art of data engineering” on page 23 focuses on the key elements of data handling, including data profiling, cleansing, reduction, transformation, enrichment, and validation, and provides insights and best practices around various data integration patterns. Since the quality of the input data can determine an AI projects' success or failure, following these methods is critical within the data understanding phase.

Model building

The model building phase is where the key AI intelligence is created. It utilizes the intersection of data science, programming and business understanding skills to leverage the significant power and capabilities of machine learning and AI. Most models contain one or more algorithms that contain logic and features which during inferencing determine how to process and interpret data and conditions present to arrive at a conclusion. In some cases, it is useful to seek insight from key experts within the business on the targeted topic area to help strengthen the algorithm. The final stage of model build involves designing and testing the model to validate it can produce the desired insights initially envisioned. In addition to the basic functionality the model must be evaluated for accuracy, consistency, and non-biased results.

Analysis of model learning quality and accuracy

Chapter 4, “Model building for IBM Z” on page 51 covers key aspects of AI model development, testing, and use. Model frameworks to facilitate model training, model formatting for AI use, and examples of and best practices for developing and executing training strategies are some of the key activities during the AI model building and deployment phases covered in this chapter.

Model deployment

Upon completion of the model build phase, it is ready to be deployed and put into service on active and eventually production workloads. AI models need to be deployed using a support AI model format that the AI inference and processing environment supports. In the early stages of model deployment, it is often advisable to follow standard test to production DevOps techniques to see the results of the model on production-like data in the application test environment before moving to production environments.

Note: DevOps is an approach to lean and agile software delivery that promotes closer collaboration between lines of business, development and IT operations. Historically, development and operations, and even testing, have been siloed operations. DevOps brings them together to improve and reduce the time needed to address customer feedback.

Advantages of co-location of data and AI applications on IBM Z

Chapter 5, “Model deployment and inferencing” on page 71 goes deeper into model deployment and inferencing and highlights the advantages that IBM Z provides in creating close data adjacency as well as the streamlined benefits of using Machine Learning for IBM z/OS for proper deployment and efficient inferencing and thus the successful integration of AI and machine learning into practical solutions.

Model monitoring

Model monitoring is a key step to maintain the health and effectiveness of the AI model. Complete and consistently applied governance techniques and processes are essential and required by law. During this phase, the AI model should be continuously measured and monitored as more, and possibly varied data are introduced to the model. There are generally two areas of focus during this phase, model accuracy, including exaplainability, and model performance.

Leveraging MLz for machine learning operations

Chapter 6, “Streamlining AI from models to applications using machine learning operations” on page 113 discusses and demonstrates the power of MLz and overall IBM Z infrastructure when looking to optimize and integrate a modern and robust machine learning operations process. A process that is essential for clients to operationalize AI on IBM Z.

1.6 Unique value of AI applications on IBM Z: data and transaction gravity

Organizations are increasingly harnessing the power of AI to gain a competitive edge. At the heart of this transformation lies the concept of the AI Ladder, illustrated in Figure 1-3, a systematic approach that paves the way for organizations to successfully integrate AI into their operations and applications. In this section, we will explore how IBM Z and the Machine Learning for IBM z/OS platform play a pivotal role in helping organizations ascend the AI Ladder and achieve AI-infused enterprise applications.

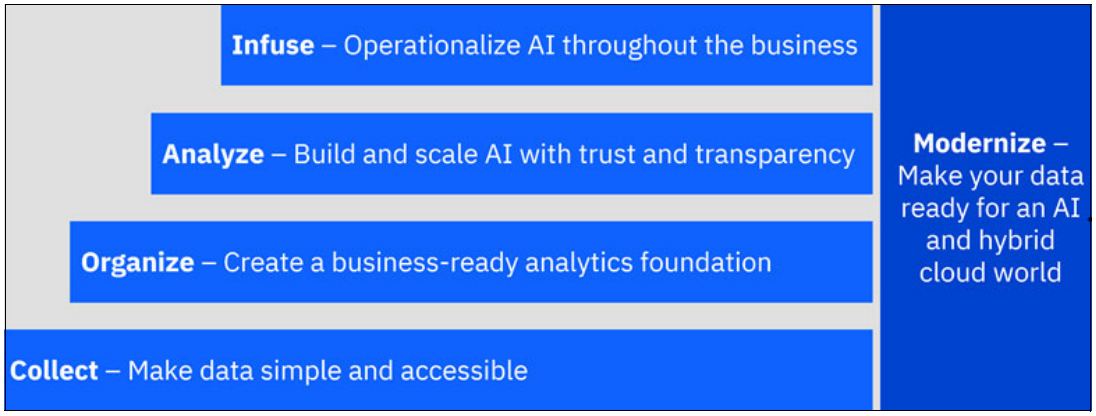


Figure 1-3 The AI ladder - A guiding strategy for organizations to transform their business by connection data and AI

The IBM AI Ladder® comprises four distinct but interconnected steps: Collect, Organize, Analyze, and Infuse. Each step builds upon the previous one, creating a comprehensive framework for implementing AI successfully. With some of the most mission-critical enterprise workloads running on IBM Z, organizations can hugely benefit from adopting the AI Ladder approach as a roadmap to accelerate their AI adoption on IBM Z platform and realize significant business value.

Another important concept to understand is the idea of data gravity and transaction gravity on IBM Z - Data gravity refers to the quantity of data originating on IBM Z and transaction gravity refers to the number of transactions going through IBM Z. Today, over an estimated 70% of the world's financial transactions runs on IBM Z.[5]

As businesses dive into expansive AI projects, they are encountering higher expenses and risk due to their initial decisions about where to set up their AI operations. Many companies that position their AI model training infrastructure far from their central data storage are facing amplified costs and heightened risk as their data collections expand and their AI models become more intricate.

Mainframe computers play a central role in the daily operations of most of the world's largest corporations. To retain the core strengths and attributes of the IBM mainframe platform and simultaneously leverage the extensive cloud services, security, and regulatory compliance programs of IBM Cloud, IBM recommends a hybrid cloud approach to mainframe application modernization. In this way, AI models can be trained within a client-chosen, security-rich environment, for example, IBM Cloud for Financial Service as discussed in Chapter 3, and the model can be deployed to IBM Z. It is critical to maintain the transaction logics and run model inference on mainframe to meet stringent low-latency, large-scale transaction requirements.

This is accredited to IBM Z platform's strong legacy of being one of the most robust, securable, and scalable enterprise platforms in the market for hosting some of the most mission-critical and core business workloads of our clients.

Expanding on the AI ladder powered by the data and transaction gravity of IBM Z

The Collect step of the AI Ladder is the foundational step at which lies the data - the lifeblood of AI. Your mission-critical data originates and resides on IBM Z. Connecting different data sources from within of IBM Z with data tools provides the rock-solid foundation needed to start the AI journey.

Once you have the data, the next step on the AI Ladder is Organize. This step helps organize the data in an optimized way to provide a business ready analytics foundation on IBM Z platform.

With the data in place, you can build and scale AI with trust and transparency. Analyze in smarter ways and benefit from AI models that empower organizations to gain new insights and make better, smarter decisions as new stream of data is continuously generated and fed to the model.

With the AI models deployed, clients can now operationalize AI at the Infuse step of the AI Ladder by infusing the models into their mission critical applications running on IBM Z drawing on predictions, automation, and optimization for real-time business insights at scale.

Machine Learning for IBM z/OS: an enterprise machine learning solution

Machine Learning for IBM z/OS is IBM's flagship machine learning platform tailored for z/OS environments that facilitates the development, deployment, management, and scaling of AI models within the secure and powerful z/OS environments in IBM Z. With some of the world's most mission-critical and transactional workloads running on z/OS environments, our clients can infuse AI into those applications by collocating their AI workloads close to those applications. By co-locating AI workloads, our clients can leverage the inherent strengths of IBM Z, while tapping into the capabilities of advanced AI and ML algorithms for high throughput and low latency inferencing to meet even the most stringent of client SLA requirements. This co-location approach addresses the growing demand for processing large-scale AI tasks efficiently, making it particularly beneficial for enterprises with vast amounts of data and complex AL and ML models.

[5] CELENT: Operationalizing Fraud Prevention on IBM z16 - Reducing Losses in Banking, Cards, and Payments, https://www.ibm.com/downloads/cas/DOXY3Q94

Unmatched capabilities of MLz

Real-time, in-transaction analytics gives companies the power to fully leverage their valuable historical data. It allows them to automatically uncover valuable insights and create predictive models that can make use of the vast amount of data they capture. Instead of relying on past reports, businesses can now predict future outcomes by analyzing their data.

Machine Learning for IBM z/OS empowers data scientists to build, train, and deploy AI models directly on IBM Z. Meaning clients can infuse AI into their mission-critical transactional applications and score every transaction in real-time on the IBM Z platform. Through doing so they can achieve accelerated business insights at scale. We will describe more about the capabilities and the breadth of use cases we support in the later chapters.

1.6.1 How MLz benefits from Telum on-chip AI accelerator

With the introduction of IBM z16 in 2022, IBM released AI capabilities at the heart of the hardware. The IBM z16 came with an industry-first on-chip AI accelerator - The IBM z16 Integrated Accelerator for AI, represents a ground breaking advancement that promises to redefine the possibilities for AI workloads in the enterprise space. This combination of cutting-edge hardware and software capabilities is poised to transform the way organizations process and analyze data, enabling them to derive business insights at unprecedented scales while meeting the rigorous demands of mission-critical workloads.

The Integrated Accelerator for AI has been designed for the sole purpose of accelerating AI workloads. It brings a new level of performance to AI inference tasks, significantly reducing the time required for model inference and enabling real-time decision-making capabilities. Its architectural design optimally aligns with the demands of AI workloads, making it a perfect companion to the IBM z16 mainframe.

This flexible on-chip Integrated Accelerator for AI works in conjunction with a standard Z-core general processor. Every general core processor in the IBM z16 system has an on-chip accelerator built-in. This centralized on-chip AI accelerator can be shared and accessed by all eight cores (see Figure 1-4). As a result, each core can access the full stack accelerator directly instead of only having access to the performance of the AI inference in each core.

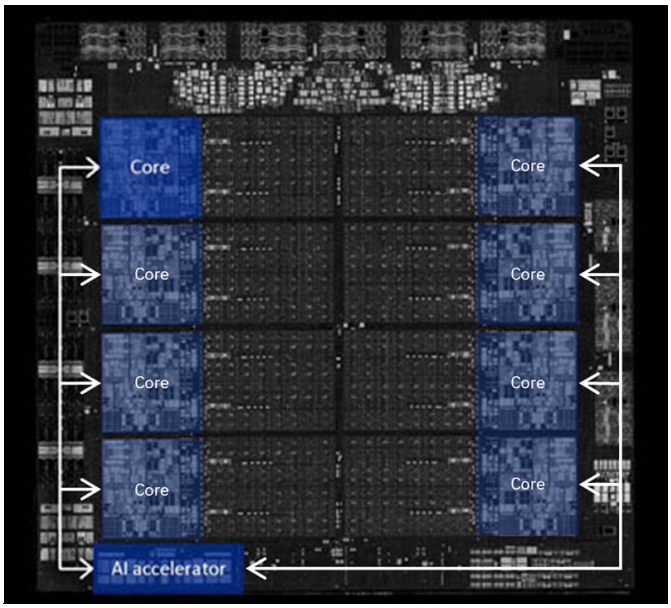


Figure 1-4 IBM Telum® On-chip Accelerator for AI

Some functional and performance features of the IBM z16 Integrated Accelerator for AI include:

- Neural network processing assist instruction - Memory-memory CISC instruction which operates directly on tensor data in user space and enables matrix multiplication, convolution, pooling, and activation functions.
- Including optimization on firmware level for core and AI accelerator to address translation and access - Address translation and access check for tensor data, pre-fetching of tensor data into L2 cache, and coordination of data staging and computing activities.
- Providing an aggregate of over 6 TFLOPS/chip (over 200 TFLOPS on a 32-chip system).
- Matrix array: 128 processor tiles with 8-way FP-16 FMA SIMD (Optimized for matrix multiplication and convolution operations).
- Activation array: 32 processor tiles with 8-way FP-16/FP-32 SIMD (Optimized for activation functions and complex operations like RELU, Sigmoid, tanh, log, high-efficiency SoftMax, LSTM and GRU).
- Intelligent pre-fetcher and writeback: 200+ GB/s read/store bandwidth from/to cache; multi-zone scratchpad for concurrent load execution.
- Intelligent Data Mover and Formatter: 600+ GB/s bandwidth between engines to format and prepare data on the fly for compute and write-back.

With more sophisticated AI algorithms being used in the industry, deep learning models (based on deep neural network) is one such subset of AI algorithms that has gained popularity and has a much-increased capacity and learning rate. Based on industry conversations and research, Celent estimates that AI inferencing based on deep learning models can increase the accuracy of fraud detection by 60% over existing fraud models. This exponentially improves throughput and response times, making it possible, for the first time, to pass virtually all transactions through deep learning-based fraud detection models.[6]

The power of using inferencing to identify and prevent fraud is limited when it comes to high-volume mainframe systems. In these situations, these models are usually used for less than 10% of transactions because of issues like slow processing speeds, high costs, and inconvenience for customers. This means that around 90% of fraud that could be prevented goes undetected. This makes it difficult for banks to use AI to reduce their losses from fraud.

But with the powerful z16 Telum on-chip AI accelerator, organizations can run AI models directly on the chip and in real-time, enabling 100% of transactions to be scored with high throughput and low latency.

With Machine learning for IBM z/OS, our clients can deploy AI models (in this case, deep learning models) by easily converting the model to ONNX format. (ONNX is an open format for representing AI models. It is open-source and vendor-neutral). Once converted, those models can be imported to MLz, deployed, and infused into our client's transactional applications for real-time AI inferencing. In the backend, the models leverage IBM Z Deep Learning Compiler (which uses ONNX-MLIR) to compile deep learning AI models in “.onnx” format into shared libraries. The shared libraries can then be integrated into C, C++, Java, or Python applications. The compiled models take advantage of IBM Z technologies including SIMD on IBM z13® and later and the Integrated Accelerator for AI available on IBM z16 without changes to the original model.

[6] CELENT: Operationalizing Fraud Prevention on IBM z16 - Reducing Losses in Banking, Cards, and Payments, https://www.ibm.com/downloads/cas/DOXY3Q94

Some performance-proof points running transactional workloads with inference operations on IBM z16 and Machine Learning for IBM z/OS v3.1 include:

- An IBM z16 system can process up to 228K z/OS CICS® credit card transactions per second with 6ms response time, each with an in-transaction fraud detection inference operation using a DL model.

  DISCLAIMER: Performance result is extrapolated from IBM internal tests running a CICS credit card transaction workload with inference operations on an IBM z16. A z/OS V2R4 LPAR configured with 6 CPs and 256 GB of memory was used. Inferencing was done with Machine Learning for IBM z/OS 2.4 running on Websphere Application Server Liberty 21.0.0.12, using a synthetic credit card fraud detection model (https://github.com/IBM/ai-on-z-fraud-detection) and the Integrated Accelerator for AI. Server-side batching was enabled on Machine Learning for IBM z/OS with a size of 8 inference operations. The benchmark was executed with 48 threads performing inference operations. Results represent a fully configured IBM z16 with 200 CPs and 40 TB storage. Results may vary.

- On IBM z16, run a CICS credit card workload with in-transaction inference operations with 55% lower response time and 119% higher throughput by leveraging the Integrated Accelerator for AI for low latency inferencing.

  DISCLAIMER: Performance results are based on an IBM internal CICS OLTP credit card workload with in-transaction fraud detection running on IBM z16. Measurements were done with and without the Integrated Accelerator for AI. A z/OS V2R4 LPAR configured with 12 CPs, 24 zIIPs, and 256 GB of memory was used. Inferencing was done with Machine Learning for IBM z/OS 2.4 running on Websphere Application Server Liberty 21.0.0.12, using a synthetic credit card fraud detection model (https://github.com/IBM/ai-on-z-fraud-detection). Server-side batching was enabled on Machine Learning for IBM z/OS with a size of 8 inference operations. Results may vary.

- On IBM z16 with z/OS, co-locating applications with inferencing requests helps minimize delays caused by network latency, delivering up to 20x lower response time and up to 19x higher throughput versus sending the same inferencing requests to a compared x86 cloud server with 60ms average network latency.

  DISCLAIMER: Performance results based on IBM internal tests using a CICS OLTP credit card workload with in-transaction fraud detection. A synthetic credit card fraud detection model was used: https://github.com/IBM/ai-on-z-fraud-detection. On IBM z16, inferencing was done with MLz on zCX. Tensorflow Serving was used on the compared x86 server. A Linux on IBM Z LPAR, located on the same IBM z16, was used to bridge the network connection between the measured z/OS LPAR and the x86 server. Additional network latency was introduced with the Linux "tc-netem" command to simulate a remote cloud environment with 60ms average latency. Results may vary.

  IBM z16 configuration: Measurements were run using a z/OS (v2R4) LPAR with MLz (OSCE) and zCX with APAR– oa61559 and APAR - OA62310 applied, 8 CPs, 16 zIIPs, and 8GB of memory. x86 configuration: Tensorflow Serving 2.4 ran on Ubuntu 20.04.3 LTS on 8 Skylake Intel Xeon Gold CPUs @ 2.30 GHz with Hyperthreading turned on, 1.5 TB memory, RAID5 local SSD Storage

- IBM z16 with the Integrated Accelerator for AI provides 4x faster response time versus IBM z15® when both are running equivalent OLTP workloads with batched fraud detection.

DISCLAIMER: Performance results based on IBM internal tests running online transaction processing (OLTP) credit card workloads with in-transaction fraud detection (https://github.com/IBM/ai-on-z-fraud-detection). On IBM z16 A01 and z15 T01, both systems ran z/OS 2.4, had 4 Central Processors, 8 z Systems® Integrated Information Processors (zIIPs) with simultaneous multithreading 2, and 16 GB memory. Inferencing was done in IBM Machine Learning for IBM z/OS Online Scoring Community Edition v1.0.0 in a single IBM z/OS Container Extensions (zCX) container. zCX was version V2R4 with APAR 0A59865. The application ran in CICS v5.4 on IBM WebSphere® Application Server Version v8.5 Liberty with Java 8.0.6.20 and IBM Enterprise COBOL for z/OS 6.2.0 P190522. The database for the application was a co-located IBM DB2® for z/OS v12. The workload driver, JMeter, was based on an initial workload that targeted 10,000 transactions per second on z15 without fraud detection. This same driver configuratoi n was then used with fraud detection on both systems where the 32 most recent transactions for that credit card were batched client-side for fraud detection.

For a full list of supported operations on the accelerator see:

https://ibm.github.io/ai-on-z-101/z16Accel/

1.6.2 Leveraging MLz in IBM Z application modernization

The mainframe is as modern and powerful as any other platform in the market. For most mission-critical applications around the world still running on mainframes, it talks to the strengths of the platform. However, most applications were designed at a time when AI was not widely used. In addition to that, applications were written in COBOL, PL/I, etc. language, and any major changes to those code bases can potentially break and disrupt the workload.

With applications running on IBM Z, our clients are increasingly looking for opportunities to modernize those applications in place. Infusing AI into these traditional applications can be complex but with MLz, our clients can modernize their applications with minimal changes to their code. This not only helps our clients extend the value of their investments but also brings more AI workloads back to IBM Z. For more of an overview of the IBM strategy to help you modernize applications faster, at lower cost and less risk, by using IBM Z and public cloud solutions together in your modernization journey, see the following IBM Redbooks Point of View publication, *Accelerate Mainframe Application Modernization with Hybrid Cloud*, REDP-5705.

IBM Z clients have the following three common ways to modernize their applications with AI and Machine Learning for IBM z/OS:

- “REST APIs”
- “CICS Exec Link Command”
- “WOLA Interface”

REST APIs

This is a widely used method for connecting different systems and services over HTTP requests. With MLz, machine learning models can be exposed as a service enabling IBM Z applications to score/infer transactions making this approach a very versatile for application modernization.

CICS Exec Link Command

Clients with CICS COBOL applications can leverage the CICS exec link command to invoke MLz services directly into the application with very minimal application changes allowing them to score/infer transactions with very high throughput and low latency.

WOLA Interface

For IMS COBOL applications, WOLA provides a high-performance bridge between IBM WebSphere for z/OS and COBOL, PL/I, C, and C++ applications in external address spaces that run on the same z/OS system by using shared memory between processes and applications. This interface can be leveraged to use the MLz scoring capability for even higher performance inferencing for large transaction volumes and low latency requirement SLAs.

2

Building business-driven and strategic use cases with MLz

Before businesses can begin to exploit Artificial Intelligence (AI) for additional value in their existing pipelines and applications, they must understand why they're implementing AI. AI is not a hammer looking for a nail. Organizations need objective criteria for evaluating business problems and pain points and determining what solutions are most appropriate. Through this section we'll dive into making sure businesses are ready for, and how to properly identify and prepare AI use cases for operationalization.

With an average of about half of AI projects making it from pilot to production, it is clear that organizations struggle to successfully operationalize AI[1]; whether that is due to lack of expertise, useful data, or a lack of business value gained in the end, we are going to show you how to avoid these pitfalls through proper planning from ideation to execution.

In “Unique value of AI applications on IBM Z: data and transaction gravity” on page 8 we discussed the AI Ladder and how its usage can aid in the implementation of AI in organizations, but the proper collection and usage of data is only one facet in the preparation of implementing AI.

Organizations must scrutinize their business plans for market opportunities and existing pain points. They must identify which opportunities and pain points may be suited to AI based on predetermined, objective criteria. Then, working backward as a check, undertake a data monetization exercise to explore existing pools of data wealth where AI can help unlock business value. This can be broken down into the following steps:

1. Planning, which includes self-assessment for organizations readiness
2. Ideation, or our brainstorming step
3. Assessment, where we go through our ideas iteratively for viability
4. Execution, where we begin MVPs and assess KPIs for our use cases

[1] https://www.ibm.com/resources/the-data-differentiator/scale-ai

2.1 Planning

Before even beginning the ideation phase, an organization needs to properly identify its strengths, maturity, and areas it needs to improve on related to AI capabilities. IBM has identified six organization AI capabilities (see Figure 2-1) that are crucial for creating good business value and return on investment (ROI) from AI projects: Vision and Strategy, Operating Model, AI Engineering and Operations, Data and Technology, Talent and Skills, and Culture and Adoption, with Trust at the center.

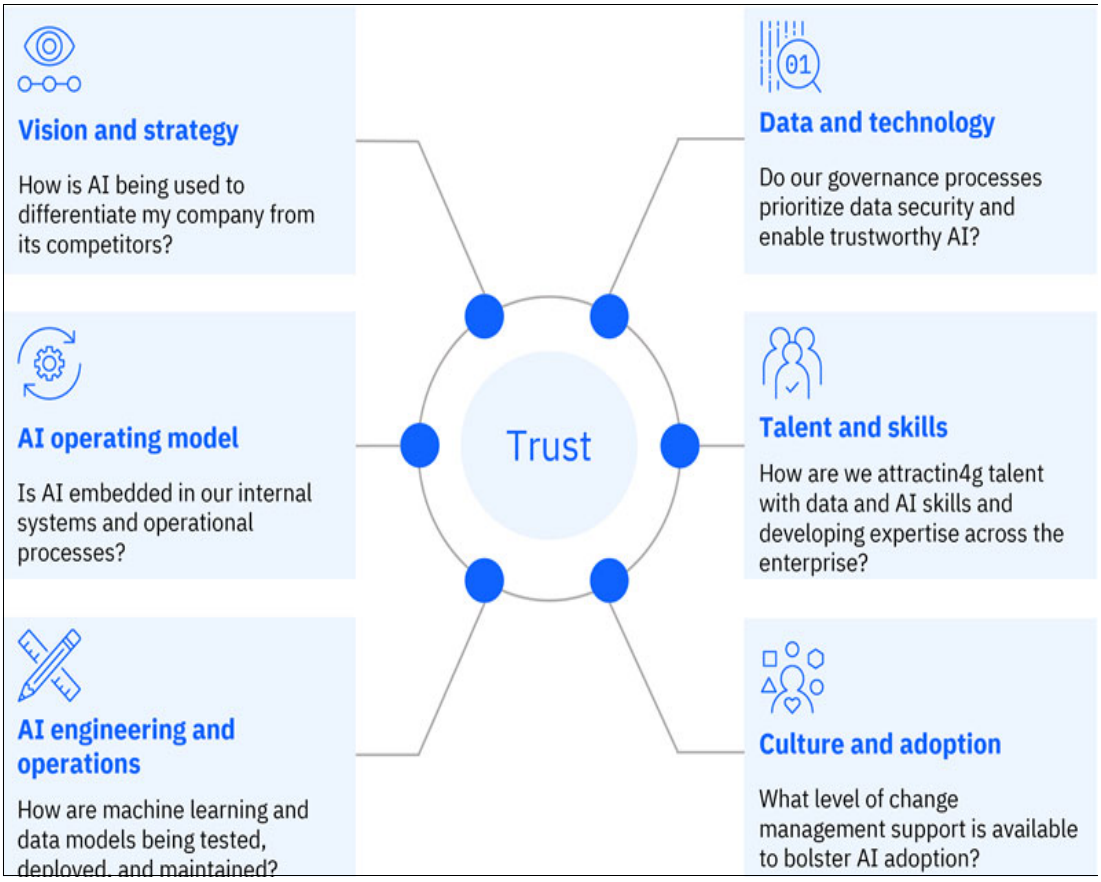


Figure 2-1 The six organization AI capabilities

Questions need to be addressed as to the maturity for each capability for your organization to understand not only how ready they are for AI, but how much they can get out of it in terms of ROI.

Vision and strategy

- What value should AI bring to your company?
- How is AI being used to differentiate your company from its competitors?
- How are AI projects reviewed for strategic alignment?
- Do you have an enterprise-wide approach to AI and a roadmap to deliver results?

Operating model

- Is AI embedded in your internal systems and operational processes?
- What is your process for developing MVPs to meet a specific business need?
- How are AI insights being generated and delivered to the business to create value?
- What checks and balances do you have in place to ensure you are using AI ethically?

AI engineering and operations

- How are machine learning and data models being tested, deployed, and maintained?
- How are you tracking changes made to AI solutions?
- Do you have systems and processes in place to identify and solve problems as they appear?
- Can you measure and tune AI models once they are deployed?

Data and technology

- Do you have high-volume, high-quality, trusted data?
- Do your governance processes prioritize data security and enable trustworthy AI?
- What level of data advocacy and literacy exists across your organization?
- Do you have the information architecture in place to scale AI solutions?

Talent and skills

- How is your organization attracting talent with data and AI skills?
- How are you developing AI skills and expertise across your organization?
- How are teams sharing knowledge to increase everyone's comfort level with AI applications?

Culture and adoption

- Is your organization change-ready?
- What level of change management support is available to bolster AI adoption?
- Do all AI projects have a named executive sponsor?
- Are KPIs baked into use case adoption?

For more information on becoming AI-ready and questions organizations can measure their AI maturity with, see the IBM Institute for Business Value's guide Generating ROI with AI:

https://www.ibm.com/downloads/cas/DDORNOB2

2.2 Ideation

For an AI solution to provide value it must target an existing problem. Teams of AI experts and strategic leaders must work together to identify business critical problems that exist within the organization and be aware of existing or future data sources that could be leveraged to tackle this problem. Organizations must scrutinize their business plans for market opportunities and existing pain points. They must identify which opportunities and pain points may be suited to AI based on predetermined, objective criteria.

Breaking down ideation into plain steps, you simply need to identify strategic pain points or opportunities for improvement, define these targets' requirements to leverage AI in this space such as data availability, and make sure that all stakeholders and parties are involved in these brainstorming steps. Something technically viable may not be strategically relevant to the business as a whole or vice versa, and having those with all levels of expertise will maximize an organization's ability to produce higher result AI projects.

2.3 Assessment and execution

After having performed the ideation, try looking back to ensure everything works, and then explore available pools of data to see how AI can create value for your business. Consider

whether you have sufficient data with which to train and inform an AI-based solution. Is the problem strategic enough to justify an investment in AI?

Maximizing business value can be found through a few general but important steps. The first of which is ideation. This is the most important step as mentioned before, for an AI solution to provide value it must target an existing problem. Find business critical problems that exist within the organization and be aware of existing or future data sources that could be leveraged to tackle this problem.

Once you have pin pointed the main issues in your organization during ideation, it is important to go through these problems, iteratively and assess the relative value solving these problems can provide. For each proposed solution identified during ideation, assess them based on ease of solution implementation, whether the necessary data is available, the impact they would have on solving business problems, and is the skills needed to implement them are available (see Figure 2-2). By taking an iterative approach, one can make sure pitfalls were not missed and ensure prioritization around solutions that maximize business value.

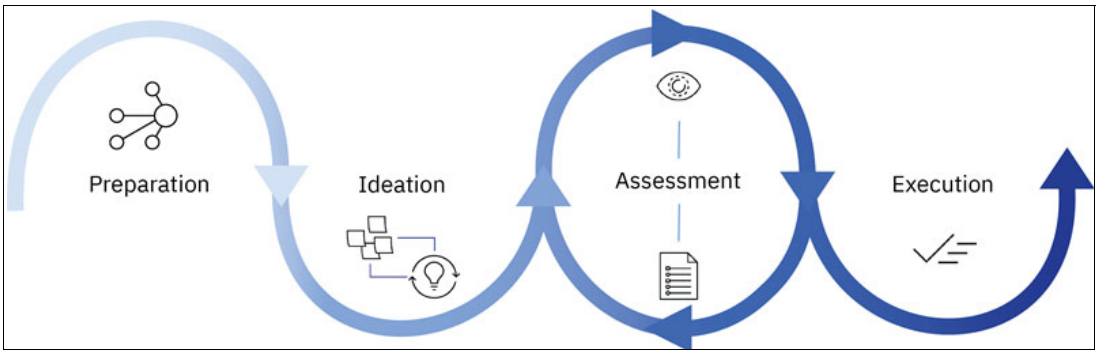


Figure 2-2 Steps involved in the ideation process

While execution itself will depend entirely on the project, it is important to understand the metrics you are collecting around this new deployment to ensure that your organization is properly maximizing business value. It is also important to always triple-check your assumptions before a costly development and deployment cycle. Go through the benefits and costs you built through ideation and assessment.

2.4 AI applications in various fields

With the knowledge on how to identify and prepare for artificial intelligence (AI) use cases, let us move on to some of the most popular or useful use cases for AI on IBM Z that we have identified.

According to the IBM Global AI Adoption Index (2022), 35% of companies reported using AI in their business, and an additional 42% reported they are exploring AI. Around half of organizations are seeing benefits from using AI to automate IT, business or network processes, including cost savings and efficiencies (54%), improvements in IT or network performance (53%), and better experiences for customers (48%).[2]

The use cases that will be shown in this chapter will show the benefits of implementing AI solutions on IBM Z.

[2] https://www.ibm.com/watson/resources/ai-adoption

2.4.1 Banking and finance

At the forefront of which is fraud detection of varying types in financial industries. It is estimated that in 2021, there were US$385 billion in global losses due to payment, card, and banking fraud. Breaking this down further, banks suffered US$328 billion in losses globally from fraud in 2021. Card and payment transaction sectors amounted to US$57 billion in losses. This is a major pain point for financial institutions and a major target for maximizing business value.[2] Figure 2-3 exemplifies the major impact and value an AI-optimized system can have in today's financial sector. With over an estimated 70% of global financial transactions' value running on IBM mainframes, being able to minimize this fraud comes more and more into focus.

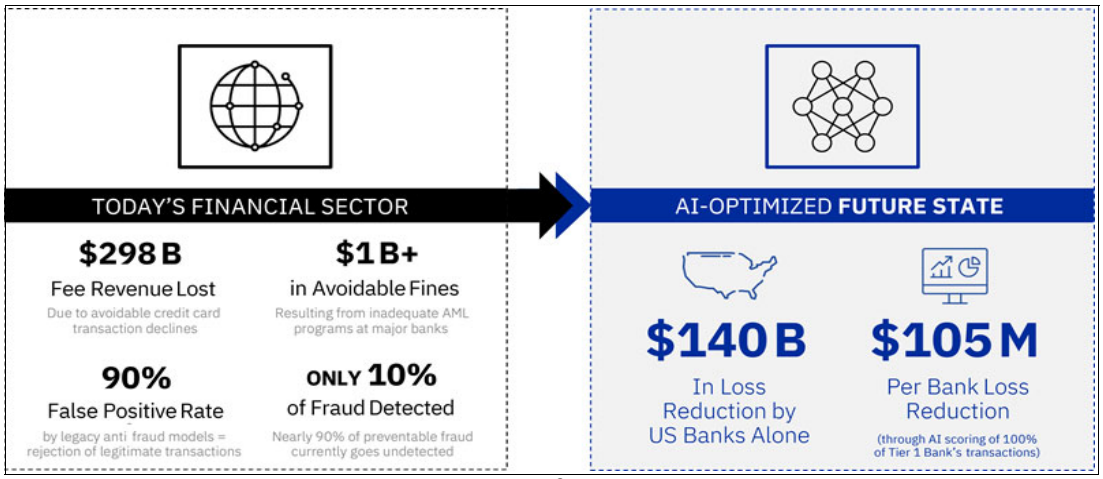


Figure 2-3 Impact of AI on today's Financial sector[3]

For card transactions, most of this fraudulent activity was from transactions where the card itself was not physically present, such as online purchases. To make matters worse, estimates show that only 10% of transactions can be tested for fraud in real-time; that is, testing for fraud at the time of the transaction.[3] One of the reasons this has been so frustrating for banks is that they already can handle these problems, but latency, cost, and custom friction have prevented them from analyzing every transaction. On IBM Z, these problems can be handled through AI leveraging deep neural networks (DNNs) and IBM z16's Telum Processor and its AI Accelerator.

Card and payment processors can reap the full potential of modern inferencing technology by running advanced models inference against all transactions. Estimates show that applying these deep learning models to all transactions across the globe would reduce fraud losses by approximately 2.0¢ cents for every $100 of transactions. This comes out to potentially reducing fraud losses by US$161 billion globally, with most, 87%, of these losses being mitigated by banks and the rest by card and payment processors.

In the US, where fraud rates are higher than the global average-9.3¢ for every $100 compared to 3.7¢ globally-fraud losses could be reduced by 5.6¢ for every $100. This is equivalent to saving the bank US$1.33 for an average transaction of US$2,375.

More information on fraud in financial services and how AI on IBM Z can solve it can be found in *Solving challenges of instant payments by using AI on IBM zSystems*, REDP-5698.

[3] CELENT: Operationalizing Fraud Prevention on IBM z16 - Reducing Losses in Banking, Cards, and Payments, https://www.ibm.com/downloads/cas/DOXY3Q94

Payment fraud detection

While deep learning algorithms and DNNs may be the solution for detecting fraud, they end up being the cause of this detection bottleneck as they tend to require more compute power than legacy fraud models. As financial institutions implement these models for fraud detection, they need to find ways to minimize inference time for calls to the model for fraud detection. For instance, round-trip time for these fraud detection inference calls can take upwards of 80 milliseconds.[4] This network latency overhead is critical when handling thousands or tens of thousands of transactions per second.

Due to these latency and throughput limitations, banks have experienced transactions timing out while they wait for detection results. This holdup is what forces these banks to only be able to screen 10% of their transactions in real-time.

In 2022, IBM released its IBM z16 mainframe computer with a processor with an integrated AI accelerator that can run AI inferencing models directly. With this innovation, the throughput and improvements of running AI models on the mainframe are sufficient to support real-time fraud analysis of virtually all transactions in even high-volume bank, card, or payments processing environments.

Moreover, this can be done with virtually no impact on transaction processing times. Our Integrated Accelerator for AI, part of our new Telum processor, can run AI models on the mainframe with response times that allow every transaction to be screened for fraud.[4]

On IBM z16 with z/OS, co-locating applications with inferencing requests helps minimize delays by network latency, delivering up to 20x lower response time and up to 19x higher throughput versus sending the same inferencing requests to a compared x86 cloud server with an average of 60ms network latency. This is even further improved with inferencing utilizing the AI Accelerator.

For one financial institution, their transactions' fraud screening needed to be completed in under 10ms, and with a throughput of 10'000 transactions per second; this could be a herculean task. Their use case was an IBM Customer Information Control System (CICS) application that handled credit card transactions with a need for fraud prevention.

The issue here was that even machines in the same data center had additional latency that made it impossible to meet performance requirements while also making AI fraud predictions for each transaction in real-time. With Machine Learning for IBM z/OS, organizations can easily deploy AI models not just on their IBM Z mainframes, but into the same CICS regions where their applications reside within seconds.

By co-locating the deep learning model not only on the same system, but in the same address space by deploying it within the same CICS region, latency for fraud detection calls are able to be reduced to the timing of function calls within the same program to meet this organization's requirements through reduced latency and increased throughput provided by the AI Accelerator.

Section 5.1.2, “Drivers for mainframe deployment” on page 74 provides greater detail around the significant advantages driving the adoption of use cases, such as fraud detection, for deployment on IBM Z.

Instant payment AML screening

Anti-money laundering (AML) efforts consist of laws, regulations, and procedures that are designed to prevent criminals from exchanging money obtained through illegal activities,

[4] CELENT: Operationalizing Fraud Prevention on IBM z16 - Reducing Losses in Banking, Cards, and Payments, https://www.ibm.com/downloads/cas/DOXY3Q94

sometimes called dirty money, into legitimate income, sometimes called clean money. Inadequate AML programs lead to over US$1 billion in avoidable fines for major banks.

In the same way that real-time transaction processing on IBM Z with MLz can prevent credit card fraud, it can be used to prevent money laundering. By allowing immediate screening of instant payments, risk can be reduced drastically.

A large bank needs to introduce AML screening into their instant payments operational flow. Their current end of day AML screening is no longer sufficient due to stricter regulations. Through the deployment of their fraud detection models through MLz onto the same Z that is handling the payments, they're able to implement stronger models for improved accuracy that met stricter regulatory requirements without impacting SLAs.

Loan approval risk and exposure

Both lenders and clients deal with onerous loan approval processing times. Through co-location of AI that handle loan risk assessment onto the same IBM Z system that processes the application, lenders can see a reduction in latency of over a thousand times, with corresponding decrease in lender risk and exposure. This leads to not only less risk for the lender, but happier clients as the process becomes easier and quicker.

Overdraft limit default prevention

Much like AML screening or credit card fraud prevention, the problem with handling over drafting of accounts for banks is a matter of timing. An overdraft is simply a deficit in an account caused by withdrawing more money than the account holds, also known as an instance of non-sufficient funds. Many financial institutions handle this by lending the difference and applying a charge for the service.

How can these institutions properly determine and handle risk for each account, though? One IBM client has dealt with it through AI on IBM Z, by augmenting their existing rules-based approach with AI to enable intelligent overdraft adjustments that reduce risk exposure on the fly.

Clearing and settlement risk and exposure

As with the trend, the issue AI solves is timing. One IBM client, a card processor, has a need for increasing the accuracy of its AI in determining which trades or transactions have a high-risk exposure before settlement.

Much like its predecessors in this list, enhancement of rules-based approaches with AI co-located on their Z system allowed them to increase the accuracy of these predictions drastically without any impact on their service level agreements. Unlike the others listed, this client leveraged deploying their TensorFlow AI models on IBM z/OS Container Extensions (zCX) rather than into CICS through MLz.

For more information about zCX, see:

https://www.ibm.com/docs/en/zos/3.1.0?topic=extensions-what-is-zos-container

Insurance claims fraud detection

The insurance industry often has excessively manual processes for determining the validity of claims. A state government realized that their process for handling claims, especially the determination of fraud, was egregiously manual and intensive as they tried to scale. Each case could take up to 40 hours to vet.

By deploying their models through MLz, which were pulled directly from IBM Cloud Pak® for Data, this state government was able to process claims and detect fraud in under one minute

per case and be able to allocate their staff to high value tasks instead, saving them both time, money, and valuable man hours.

2.4.2 Other use cases

How businesses leverage AI with MLz and z/OS in other fields. While some of the more high-profile use cases for MLz and the z16 have been around the financial industry, many other industries such as health care and government agencies have been taking great advantage as well.

Automated land surveying

Most of these use cases involve complex analysis of structured data. One alternative use, is AI for image analysis. A land registry needed to leverage AI on IBM Z to analyze satellite images to determine which buildings have been added to, modified, or demolished for land surveys and tax purposes. They also deployed models for embedding natural language processing (NLP) into chat services.

By deploying their services and AI on IBM z16, this organization was able to ensure security of data as well as optimization of IT and operational costs.

Climate change impact

Along the same vein, an environmental agency was seeking to understand the impact of climate change on local coastal ecosystems. Their field agents were collecting images that needed to be analyzed for change over time. Rather than manual analysis, they opted for AI on the IBM z16. Deploying an AI service set to scale, they met their goal of automated image analysis and decreased their energy demands in the process.

System log anomaly detection

The fast pace of digitization and data explosion is generating data at a rapid rate. Navigating this large set of data explosion need a highly complex system to leverage them. Complex system management requires complex skills. The current complex system is in reactive mode, where when problems happen, the team works to solve or fix them. Many time, quick fix is utilized due to the level of risk. AI and Machine learning help organizations move from reactive mode to proactive mode. AI and ML monitor the application performance and detect before they are supposed to happen. AI and ML model goes through a large set of system logs and detect the issue before it happens. Anomaly detection combs through a large set of data to detect the signal of any anomaly. The Anomaly algorithm allows the application to see the issue and present it to the application administrator to act on it.

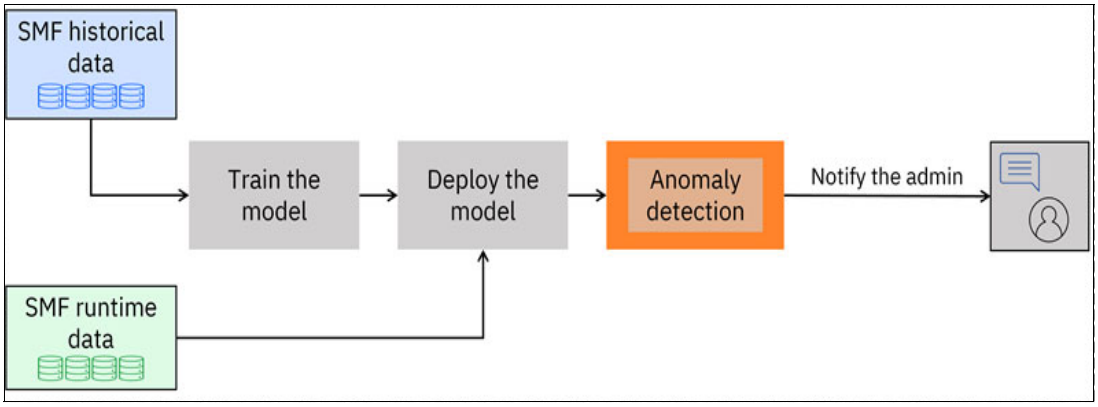


Figure 2-4 Flow for anomaly detection based on system generated logs

3

The art of data engineering

Data are important assets. Unleashing the power of transactional data and turning it into valuable insights and supporting new business opportunities can bring businesses competitive advantages. This chapter discusses the different characteristics of data, process of understanding the data, and the various patterns to make enterprise data on mainframe available for various consumptions on-premises or in cloud.

3.1 Nature of data

Data is the lifeblood of every modern organization, and it is being created, stored, and analyzed at an unprecedented rate across industries. By 2025, global data creation is estimated to reach a staggering 180 zetabytes.[1]

Data comes in great volume, velocity, and variety every day. Being able to curate the volume of data and draw insights from the data brings enterprises competitive advantages adequately and efficiently.

Much like individuals learning from past experiences, adapting to the present, and shaping the future, the AI system can learn from historical data to improve day-to-day decision-making along with future guidance. Data is the fuel to AI systems, and good quality data results in more efficient and better performing AI systems.

Real-world raw data comes in all shapes and sizes from different sources and cannot be fed to AI systems as is. Raw data could have missing entries and contain fake or even hate, abuse, and profanity (HAP) information. Raw data could also contain proprietary or sensitive information that are prohibited by law to use. Data needs to be preprocessed before feeding to AI model. We will discuss the data preprocessing steps and techniques in this chapter.

3.2 Characteristics of data

Data comes in all shapes and sizes and is being produced at exponential rates. It is critical to the success of your AI project that you find the right data set for your given task. To do so, we suggest asking a series of questions that will help you identify the most suitable data set that aligns with your objectives. Consider the following questions:

- What is the business problem at hand, and what is the desired outcome?
- What data is needed and what relevant data sources can be accessed?
- What is the data volume, velocity, variety, and veracity of the available data sources?
- What are the constraints and limitations of the data sets, such as security, compliance, governance, and auditing requirements?
- How much data is required to train and evaluate the model?
- Does the project data set require data labels or annotations?
- Does the problem benefit from a diverse data set, or are there specific characteristics needed?
- What preprocessing steps are needed to prepare the data for analytics and ML?
- Are the necessary resources available to assist with data collection and preparation?

Data classification is the process of separating and organizing data into different categories based on their characteristics. Let's take a look at the different characteristics of data.

Data can be classified based on the level of sensitivity or confidentiality to your organization. Data sensitivity classification is important for risk management and regulatory compliance. Once data sensitivity classification scheme is defined, secure data handling policies should be identified for each category. You can define your own classification. The following are some common categories of data:

[1] https://www.ibm.com/downloads/cas/EO24ZZZ4

- Public data is available to the public. It can be distributed and not sensitive in nature. Examples include company name and address, and product documentation.
- Private data is protected and not available to the general public. Compromised private data can pose a risk to an individual or an organization. Examples include personal phone number or employee id.
- Internal data is organization's data and access is limited to its employees. Examples include internal websites and server IPs.
- Confidential data is sensitive data only available to authorized personnel. The loss of confidential data is harmful to an individual or an organization. Examples include social security number and employee records.
- Restricted data is highly sensitive data that access is strictly controlled. The loss of restricted data could lead to a huge loss to an individual or an organization. Examples include trade secrets and financial records.

Data with different sensitivity levels should be handled accordingly. For example, Personally Identifiable Information (PII) is private, and it can be used individually or jointly with other information to identify a single person. Because of its sensitive nature, numerous laws worldwide regulate how PII can be stored and processed, including General Data Protection Regulation (GDPR) from Europe, Health Insurance Portability and Accountability Act (HIPAA), Gramm-Leach-Bliley Act (GLBA) and others from US, Personal Information Protection and Electronic Documents Act (PIPEDA) from Canada, and so on.

Data can be of the following different forms:

- Structured data is generally in tabular form whose fields contain data of a predefined format. Some fields might have a strict format, such as phone numbers or addresses, while other fields can have variable-length text strings, such as names or descriptions.
- Unstructured data is a compilation of various types of data stored in their native formats, such as documents, videos, audios, and social media posts.
- Semi-structured data is not in tabular form but still has some some organizational properties to it, such as XML or JSON files.

Structured data is normally stored in Relational Database and is easy to organize, query and analyze, but may take more space. Unstructured data are stored in native formats and could be hard to organize and analyze but could be space optimal. Semi-structured data can be organized and analyzed with some processing.

Data can be of the following different types:

- Numerical Data is data which consists of numerical values. Numerical data can be of two types:
  - Discrete Data is countable numerical data, such as the number of products in inventory, the number of students in a class, etc.
  - Continuous Data is data that can be continually measured, such as the height or weight of a person.
- Categorical data represents distinct categories and is used to label data into specific, non-numeric data categories, which can be stored and retrieved based on the name and labels given to them. Categorical data can be of two types:
  - Ordinal data is a type of categorical data that is categorized in a way that has natural, or meaningful order or ranking among them. It is categorical data ranked by specific attributes-for example, level of education (e.g., high school, bachelor's, master's), economic status (e.g., lower income, middle income, higher income), or customer satisfaction ratings (e.g., very dissatisfied, neutral, very satisfied).

- Nominal data is a type of categorical data that is categorized in a way that does not imply any inherent order or ranking among them. It is used to categorize and label data without implying quantitative relationships between the categories-for example, eye color (e.g., blue, green, brown, hazel, gray), marital status (e.g., single, married, divorced, widowed), or car makes and models (e.g., Ford, Honda, Toyota, Camry, Civic, Focus).

For each field in a data set, understanding the data type of the field and its value range and distribution is critical for data analysis tasks.

Data could be stored in different data sources in various formats, for example, data could be saved in tabular format in Relational database (e.g., IBM DB2), JSON format in non-SQL database (e.g., IBM Cloudant®), sequential data set (e.g., VSAM) on mainframe, parquet and open table format (e.g., Apache Iceberg) in Cloud Object Storage. Data could reside at different locations, for example, on-premises, in cloud, or IoT or sensor data from edge or client locations. AI project very often needs to connect or extract data from various sources.

There are also data localization laws that enforce how data can be processed in a certain territory and data sovereignty laws around control and storage of data. Companies who handle data need to understand these laws and make sure they are compliant.

Training AI model normally requires lots of data. Depending on the model, large volume of annotated and labeled data may be needed, which may require lots of human pre-processing of the data sets. A new emerging trend in generative AI is to build general purpose foundation models which use self-supervised learning to create labels from input data. Foundation models can then be further fine-tuned with less data for downstream domain specific tasks or use prompt engineering to guide the model to generate output for various tasks. Foundation model may require more computational power depending on the size of the model. Customer care assistant and code assistant are popular foundation model use cases nowadays.

In summary, there are different characteristics associated with data sets. The ultimate goal is to generate business insights and help decision making with the various data sets. One needs to start with the business problem at hand and choose the right data sets to address the business problem. Besides the characteristics of the data sets, one also needs to take laws and regulations into consideration when handling and storing data sets.

3.3 Data preprocessing

Data is the nourishment that fuels an AI model's development, growth, and overall ability to perform effectively. It is essential that such raw data undergo data preprocessing before being used in a model. Data preprocessing cleans and transforms that raw data into a format that can be more easily and effectively used by the model, maximizing the quality and relevance of the data for AI, and enabling the model to thrive and achieve optimal performance. At a high level, as shown in Figure 3-1 on page 27, data preprocessing consists of the following steps:

1. Data profiling - A process for analyzing and understanding the structure, content, relationships, and quality of existing raw data sets and enables informed decision making for cleaning, transforming, and preparing the data set for the AI model. This process produces a high-yield overview of data quality issues, risks, and overall data patterns and trends. Data profiling is a critical and foundational part of the data preparation process for curating quality data sets for use in AI.
2. Data cleansing - Data cleansing is the process where data is explored to identify and correct any inaccuracies, corruptions, misformatting, duplicate records, missing values, or

other inconsistencies in the data. The purpose of data cleansing is to ensure data is accurate, complete, and reliable for use in AI.

3. Data reduction - The goal of data reduction is to simplify the data set while preserving the integrity and quality of that data. The data reduction process employs various optimization techniques, such as feature selection or sampling, to reduce the volume of data and help improve computational efficiency and reduce noise.
4. Data transformation - The process of data transformation involves changing the format, structure, or values of the data to be more suitable for the specific task at hand and leverage for use in producing the ML/AI models to derive insight from the given data.
5. Data enrichment - The process of data enrichment involves enhancing raw data by adding information and context to improve the relevance, value, and informativeness of the data for analysis or ML tasks. Some key aspects of the data enrichment process include integrating data from external sources, including third-party and other sources, that complement the existing data set, combining data from multiple sources for a more comprehensive data set, and feature engineering. Enriched data sources produce valuable data assets, with significantly improved quality and utility, thus providing added context and insights for better decision-making or training of AI/ML models.
6. Data validation - Data validation is the process of verifying and validating that the preprocessed data is accurate, consistent, complete, and fit for the intended modeling task before it is used for analysis or model building. This process helps identify errors, anomalies, or other issues with the data before proceeding in further analysis, a critical element to producing accurate and trustworthy results.

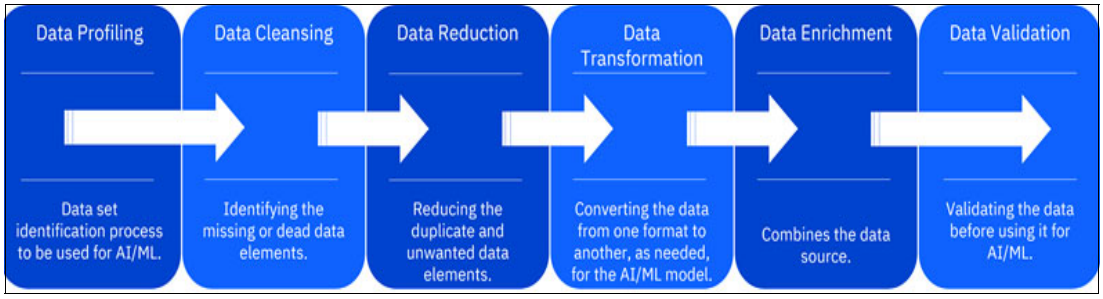


Figure 3-1 Process of understanding the nature of data

Many data scientists use Python, because of the extensive libraries available for the various phases and tasks of the AI project. Some of the popular Python libraries used for data preprocessing include NumPy, Pandas, Scikit-learn, and Matplotlib, which provide various functions and methods to perform data preprocessing tasks efficiently and effectively.

In section 1, we discussed a credit card fraud detection use case and illustrated how IBM z16 allows in-transaction model inference by leveraging the Integrated Accelerator for AI. The synthetic credit card fraud detection model along with the synthetic data set used to train and test the model can be found at the following github site,

`https://github.com/IBM/ai-on-z-fraud-detection`

In this section, we will use the synthetic credit card fraud detection data set to demonstrate some of the data preprocessing steps using Python libraries.

IBM also provides various tool to help with the data preprocessing tasks. The data refinery tool, available with IBM Watson® Studio and IBM Watson Knowledge Catalog both on-prem and in cloud, saves data preparation time by quickly transforming large amounts of raw data into consumable, high-quality information that is ready for analytics.

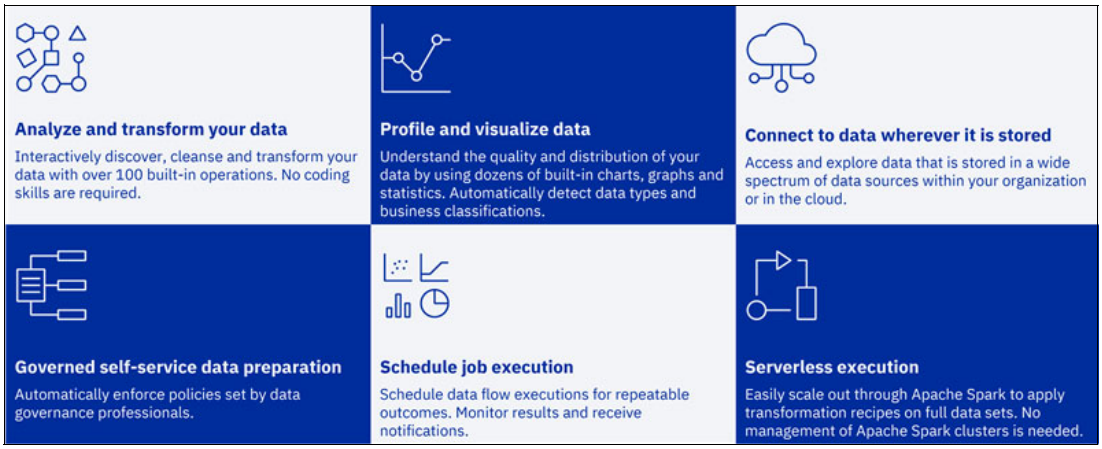


Figure 3-2 IBM data refinery features

For more information on the data refinery tool available in IBM watsonx.ai AI Studio and IBM Watson Knowledge Catalog, see:

https://www.ibm.com/products/data-refinery

3.3.1 Data profiling

Data profiling is the process of analyzing and understanding the structure, content, relationships, and quality of existing raw data sets. The main purpose is to gain insights into the characteristics and quality of the data by using methods to review and summarize it, and then evaluating its condition.

Data profiling is an essential part of how an organization handles its data. It not only can help you understand your data, but it can also verify that your data is up to standard statistical measure.

The following are approaches analysts may use to profile your data[2]:

- Structure discovery - This approach focuses on the format of the data and ensuring it is consistent all throughout the database. There are a number of different processes analysts might use for this type when examining the database. One is pattern matching, which can help you to understand format-specific information. An example of this is if you are lining up phone numbers and one has a missing value. This is something that could be caught in structure discovery.
- Content discovery - This type is when you analyze data rows for errors or systemic issues. This process is a closer look at the individual elements of the database and can help you find incorrect values.
- Relationship discovery - This type entails finding out what data is in use and trying to find the connection between each set. To do this, analysts will begin with metadata analysis to figure out what the relationships are between data and then narrow down the connections between specific fields.

The following are methods and techniques for data profiling:

- Column profiling - This method scans tables and counts the number of times each value shows up within each column. Column profiling can be useful in finding frequency distribution and patterns within a column.

[2] https://www.ibm.com/topics/data-profiling

- Cross-column profiling - This technique is made up of two processes: key analysis and dependency analysis. The key analysis process looks at the array of attribute values by scouting for a possible primary key. While the dependency analysis process works to identify what relationships or patterns are embedded within the data set.
- Cross-table profiling - This technique uses key analysis to identify stray data. The foreign key analysis identifies orphaned records or general differences to examine the relationship between column sets in different tables.
- Data rule validation - This method assesses data sets against established rules and standards to verify that they are in fact following those predefined rules.
- Key Integrity - Ensuring keys are always present in the data and identifies orphan keys, which can be problematic.
- Cardinality - This technique checks relationships such as one-to-one and one-to-many, between data sets.
- Pattern and frequency distribution - This technique ensures data fields are formatted correctly.

Take the sample data set for credit card fraud detection, for example, you can use Python libraries to analyze the data. You can load the data into a pandas DataFrame with pandas library as shown in Figure 3-3.

```
[2]: # Read data file.
import pandas as pd

# Data file could be from different sources, eg. databases, file system, Cloud Object Storage, etc.
csv_file_name = '../data/ccf_raw/card_transaction.v1.csv'
tdf = pd.read_csv(csv_file_name)

tdf.head()
```

[2]:

	User	Card	Year	Month	Day	Time	Amount	Use Chip	Merchant Name	Merchant City	Merchant State	Zip	MCC	Errors?	Is Fraud?
0	0	0	2002	9	1	06:21	$134.09	Swipe Transaction	3527213246127876953	La Verne	CA	91750.0	5300	NaN	No
1	0	0	2002	9	1	06:42	$38.48	Swipe Transaction	-727612092139916043	Monterey Park	CA	91754.0	5411	NaN	No
2	0	0	2002	9	2	06:22	$120.34	Swipe Transaction	-727612092139916043	Monterey Park	CA	91754.0	5411	NaN	No
3	0	0	2002	9	2	17:45	$128.95	Swipe Transaction	3414527459579106770	Monterey Park	CA	91754.0	5651	NaN	No
4	0	0	2002	9	3	06:23	$104.71	Swipe Transaction	5817218446178736267	La Verne	CA	91750.0	5912	NaN	No

Figure 3-3 Profiling example - load data

You can use Pandas DataFrame to analyze the characteristics of the data, for example, mean, max, count, standard deviation, and so on.

```
[3]: tdf.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24386900 entries, 0 to 24386899
Data columns (total 15 columns):
 #   Column          Dtype
---  ------          -----
 0   User            int64
 1   Card            int64
 2   Year            int64
 3   Month           int64
 4   Day             int64
 5   Time            object
 6   Amount          object
 7   Use Chip        object
 8   Merchant Name   int64
 9   Merchant City   object
 10  Merchant State  object
 11  Zip             float64
 12  MCC             int64
 13  Errors?         object
 14  Is Fraud?       object
dtypes: float64(1), int64(7), object(7)
memory usage: 2.7+ GB
```

```
[4]: tdf.describe()
```

[4]:

	User	Card	Year	Month	Day	Merchant Name	Zip	MCC
count	2.438690e+07	2.438690e+07	2.438690e+07	2.438690e+07	2.438690e+07	2.438690e+07	2.150876e+07	2.438690e+07
mean	1.001019e+03	1.351366e+00	2.011955e+03	6.525064e+00	1.571812e+01	-4.769230e+17	5.095644e+04	5.561171e+03
std	5.694612e+02	1.407154e+00	5.105921e+00	3.472355e+00	8.794073e+00	4.758940e+18	2.939707e+04	8.793154e+02
min	0.000000e+00	0.000000e+00	1.991000e+03	1.000000e+00	1.000000e+00	-9.222899e+18	5.010000e+02	1.711000e+03
25%	5.100000e+02	0.000000e+00	2.008000e+03	3.000000e+00	8.000000e+00	-4.500543e+18	2.837400e+04	5.300000e+03
50%	1.006000e+03	1.000000e+00	2.013000e+03	7.000000e+00	1.600000e+01	-7.946765e+17	4.674200e+04	5.499000e+03
75%	1.477000e+03	2.000000e+00	2.016000e+03	1.000000e+01	2.300000e+01	3.189517e+18	7.756400e+04	5.812000e+03
max	1.999000e+03	8.000000e+00	2.020000e+03	1.200000e+01	3.100000e+01	9.223292e+18	9.992800e+04	9.402000e+03

Figure 3-4 Profiling example – Pandas DataFrame

You can use Matplotlib or Seaborn library to visualize the data, or you can use Pandas profiling package, as shown in Example 3-1 (`ydata-profiling`), to automate and generate detailed reports, statistics, and visualizations.

Example 3-1 Using Pandas profiling package to automate and generate detailed reports, statistics, and visualizations

```
import ydata_profiling as pf
# sample command to generate profiling reports on screen
pf.ProfileReport(tdf)

# You can also save the profiling reports to files
profile = pf.ProfileReport(tdf, title="CCF Profiling Report")
profile.to_file('my_ccf_report.html') # Trigger the computation / alternative you
can use profile.to_json() for no file output
profile_dump('my_ccf_report') # Serialize in pickle to my_report.pp
```

As you can see in Figure 3-5 on page 31, Pandas profiling generates various reports on the variables in the data set, along with variable interactions and correlations. Taking our sample fraud detection data set for example, when loading the data as-is, the 'Amount' variable has the leading '$' sign and is treated as text. Variable 'Use Chip' has 3 distinct categories (Swipe Transaction, Chip Transaction, and Online Transaction) and has correlations with other variables. There are also some missing data in the data set.

Figure 3-5 Sample output from the Pandas profiling reports

3.3.2 Data cleansing

Data cleansing is the process to identify and correct any inaccuracies, corruptions, misformatting, duplicate records, missing values, or other inconsistencies in the data. The purpose of data cleansing is to ensure data is accurate, complete, and reliable for use in AI.

Here are some typical data cleansing tasks:

- Fix structural errors - This includes fixing inconsistent naming convention (e.g., N/A vs. Not Applicable), inconsistent capitalization (e.g., Male, male, and MALE), typos, extra spaces, or nonprinting characters in the values.
- Fix formatting issues - There may be inconsistent formatting, (e.g., format of date/time, phone number, or SSN may not be consistent), number and number format may need to be fixed.
- Handle outliers - There may be outliers in the data set, however, outlier may have legitimate reason to stay. Validate the outlier and see if it is a mistake or irrelevant for the analysis.
- Remove duplicates - Remove duplicate records from the data set.
- Handle missing data - There are different ways to handle missing data, each may have pros and cons, including removing the observations with missing data or replacing them with other values.

When we did the data profiling in the last section, we noticed that variable 'Errors?' has lots of missing values. Let us take a closer look at the missing values and see if further processing is needed.

```
# get unique values
tdf['Errors?'].unique()
```

```
array([nan, 'Technical Glitch,', 'Insufficient Balance,', 'Bad PIN,',
       'Bad PIN,Insufficient Balance,', 'Bad Expiration,',
       'Bad PIN,Technical Glitch,', 'Bad Card Number,', 'Bad CVV,',
       'Bad Zipcode,', 'Insufficient Balance,Technical Glitch,',
       'Bad Card Number,Insufficient Balance,',
       'Bad Card Number,Bad CVV,', 'Bad CVV,Insufficient Balance,',
       'Bad Card Number,Bad Expiration,', 'Bad Expiration,Bad CVV,',
       'Bad Expiration,Insufficient Balance,',
       'Bad Expiration,Technical Glitch,',
       'Bad Card Number,Bad Expiration,Technical Glitch,',
       'Bad CVV,Technical Glitch,', 'Bad Card Number,Technical Glitch,',
       'Bad Zipcode,Insufficient Balance,',
       'Bad Zipcode,Technical Glitch,',
       'Bad Card Number,Bad Expiration,Insufficient Balance,'],
      dtype=object)
```

```
# Counting NaN values in the 'Errors?' column
nan_count = tdf['Errors?'].isna().sum()

print(nan_count)
```

```
23998469
```

Figure 3-6 Data cleansing – unique values

At first glance, it seems that the column captures various error scenarios, and the missing value should be transactions without error. We can leave it as-is, or we can use the following statements to replace the missing values if needed.

```
tdf['Errors?'].fillna('Normal transaction', inplace=True)
tdf.head()
```

	User	Card	Year	Month	Day	Time	Amount	Use Chip	Merchant Name	Merchant City	Merchant State	Zip	MCC	Errors?	Is Fraud?
0	0	0	2002	9	1	06:21	$134.09	Swipe Transaction	3527213246127876953	La Verne	CA	91750.0	5300	Normal transaction	No
1	0	0	2002	9	1	06:42	$38.48	Swipe Transaction	-727612092139916043	Monterey Park	CA	91754.0	5411	Normal transaction	No
2	0	0	2002	9	2	06:22	$120.34	Swipe Transaction	-727612092139916043	Monterey Park	CA	91754.0	5411	Normal transaction	No
3	0	0	2002	9	2	17:45	$128.95	Swipe Transaction	3414527459579106770	Monterey Park	CA	91754.0	5651	Normal transaction	No
4	0	0	2002	9	3	06:23	$104.71	Swipe Transaction	5817218446178736267	La Verne	CA	91750.0	5912	Normal transaction	No

```
# get unique values
tdf['Errors?'].unique()
```

```
array(['Normal transaction', 'Technical Glitch,', 'Insufficient Balance,',
       'Bad PIN,', 'Bad PIN,Insufficient Balance,', 'Bad Expiration,',
       'Bad PIN,Technical Glitch,', 'Bad Card Number,', 'Bad CVV,',
       'Bad Zipcode,', 'Insufficient Balance,Technical Glitch,',
       'Bad Card Number,Insufficient Balance,',
       'Bad Card Number,Bad CVV,', 'Bad CVV,Insufficient Balance,',
       'Bad Card Number,Bad Expiration,', 'Bad Expiration,Bad CVV,',
       'Bad Expiration,Insufficient Balance,',
       'Bad Expiration,Technical Glitch,',
       'Bad Card Number,Bad Expiration,Technical Glitch,',
       'Bad CVV,Technical Glitch,', 'Bad Card Number,Technical Glitch,',
       'Bad Zipcode,Insufficient Balance,',
       'Bad Zipcode,Technical Glitch,',
       'Bad Card Number,Bad Expiration,Insufficient Balance,'],
      dtype=object)
```

```
# Counting NaN values in the 'Errors?' column
nan_count = tdf['Errors?'].isna().sum()

print(nan_count)
```

```
0
```

Figure 3-7 Data cleansing – replace text values

The following are sample statements to replace numerical values with median:

```
median_value = mydataset['column_name'].quantile(0.5)
mydataset['column_name'] = mydataset['column_name'].fillna(median_value)
```

Pandas DataFrame also provides method to drop duplicate records. For example, Figure 3-8 on page 34 is an example to get unique user and card combinations.

```
# Get first of each User-Card combination
first = tdf[['User','Card']].drop_duplicates()
first
```

	User	Card
0	0	0
5011	0	1
6214	0	2
10546	0	3
19937	0	4
...	...	...
24357917	1997	1
24373125	1997	2
24376357	1998	0
24382139	1999	0
24382226	1999	1

6139 rows × 2 columns

Figure 3-8 Data cleansing – drop duplicates

You can preprocess the data with various Python libraries as demonstrated above, or you can use `DataFrameMapping` module from scikit-learn library to preprocess the data in a more clean and robust way, which we will discuss more in the data transformation subsection.

3.3.3 Data reduction

Data reduction reduces the size of a data set while preserving the most important information. It reduces the volume of data and improves computational efficiency.

There are the following data reduction techniques:

- Dimensionality reduction - Dimensionality reduction is the process of reducing the number of dimensions in a data set while retaining as much information as possible. High-dimensional data often poses more challenges and complications, including model performance deterioration, data sparsity, and overfitting issues, which is commonly referred to as the curse of dimensionality. Dimensionality reduction can help to mitigate these problems by reducing the complexity of the model and improving its performance. Different techniques can be used for dimensionality reduction, including:
  - Feature selection is to select a subset of the original features most relevant to the problem at hand. The goal is to maintain the most important features of the data set while reducing the number of dimensions. Filtering (select best features) and wrapping (try out reduced feature-spaces) are two general techniques for feature selection. The classes in the `sklearn.feature_selection` module can be used for feature selection and dimensionality reduction. For more information on the `sklearn.feature_selection` module and examples for the various techniques, see:

    `https://scikit-learn.org/stable/modules/feature_selection.html`
  - Feature extraction is to create new features by combining or transforming the original features. The goal is to capture the essence of the original data set in a

lower-dimensional space by creating a set of new features. Principal component analysis (PCA) is a popular technique for dimensionality reduction. Other methods include linear discriminant analysis (LDA), t-distributed stochastic neighbor embedding (t-SNE) and auto encoders. The `sklearn.feature_extraction` module can be used to extract features. For more information on the `sklearn.feature_extraction` module and examples for the various techniques, see:

https://scikit-learn.org/stable/modules/feature_extraction.html

- Data sampling - Data sampling is the technique to select a subset of the data set, rather than using the entire data set. There are different data sampling techniques, including simple random sampling, systematic sampling, stratified sampling, and cluster sampling.
- Data discretization - Data discretization is the process of converting continuous data into discrete data by partitioning the range of possible values into contiguous intervals. Binning, histogram analysis, cluster analysis, decision tree analysis, correlation analysis are typical methods of data discretization.

Figure 3-9 is an example of random data sampling using `pandas.DataFrame.sample`.

```
tdf2 = tdf.sample(n=10, random_state=1)
tdf2
```

	User	Card	Year	Month	Day	Time	Amount	Use Chip	Merchant Name	Merchant City	Merchant State	Zip	MCC	Errors?	Is Fraud?
4866429	413	1	2011	3	15	07:50	$50.43	Swipe Transaction	-3220758452254689706	Spokane	WA	99208.0	5311	Normal transaction	No
5124282	433	1	2015	9	20	05:30	$53.78	Online Transaction	-2088492411650162548	ONLINE	NaN	NaN	4784	Normal transaction	No
17887935	1442	3	2020	2	10	11:41	$30.98	Chip Transaction	-4113349227963201766	Lakeland	FL	33813.0	5411	Normal transaction	No
7224963	610	2	2017	10	13	18:32	$10.32	Swipe Transaction	6913268435708117971	Albuquerque	NM	87121.0	5912	Normal transaction	No
19812059	1606	1	2009	5	18	10:44	$130.45	Swipe Transaction	4937803722023861373	Stockton	CA	95210.0	5310	Normal transaction	No
11960960	987	0	2019	1	6	06:08	$65.42	Chip Transaction	6181510752141427450	Avoca	MI	48006.0	7538	Normal transaction	No
17818383	1437	1	2013	12	27	14:45	$104.29	Swipe Transaction	-5467922351692495955	Brooklyn	NY	11222.0	5912	Normal transaction	No
12724868	1043	1	2018	8	18	12:09	$44.24	Online Transaction	7035602569409149834	ONLINE	NaN	NaN	5311	Normal transaction	No
5643279	484	1	2014	4	10	08:14	$23.36	Online Transaction	-6160036380778658394	ONLINE	NaN	NaN	4121	Normal transaction	No
13196474	1081	4	2017	9	16	11:57	$17.17	Chip Transaction	4751695835751691036	Nashville	TN	37211.0	5814	Normal transaction	No

Figure 3-9 Data reduction – random sampling

3.3.4 Data transformation

Data transformation is the process of converting the format, structure, or values of the data to be more suitable to be analyzed to derive insights from the data and support decision making processes.

There are many different data transformation techniques that one can choose to apply to the data set at hand, and different types of data may require different types of transformation. The following are some sample data transformation techniques:

- Format conversion
- Data normalization and standardization
- Feature construction and value mapping
- Data aggregation
- Data splitting
- Data integration

Data transformation can work hand-in-hand with the other data cleansing and data reduction techniques, for example, data transformation script could handle missing data to construct new features.

Take the sample credit card fraud detection model for example, `sklearn_pandas.DataFrameMapper` is used to transform the column values. As you can see from Figure 3-10, column Amount is first applied `FunctionTransformer` with the method `amtEncoder`, which removes the first character '$', converts the type to float, and applied log on the value, then the `MinMaxScaler` is applied to convert the value to the range of 0 to 1.

Define custom mapping functions

```
def timeEncoder(X):
    X_hm = X['Time'].str.split(':', expand=True)
    d = pd.to_datetime(dict(year=X['Year'],month=X['Month'],day=X['Day'],hour=X_hm[0],minute=X_hm[1])).astype(int
    return pd.DataFrame(d)

def amtEncoder(X):
    amt = X.apply(lambda x: x[1:]).astype(float).map(lambda amt: max(1,amt)).map(math.log)
    return pd.DataFrame(amt)

def decimalEncoder(X,length=5):
    dnew = pd.DataFrame()
    for i in range(length):
        dnew[i] = np.mod(X,10)
        X = np.floor_divide(X,10)
    return dnew

def fraudEncoder(X):
    return np.where(X == 'Yes', 1, 0).astype(int)
```

Fit the whole data using DataFrameMapper and pickle the mapper

```
save_dir = 'saved_models/P/ccf_220_keras_lstm_static/1'
```

```
from sklearn_pandas import DataFrameMapper
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import FunctionTransformer
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelBinarizer
from sklearn.impute import SimpleImputer

mapper = DataFrameMapper([('Is Fraud?', FunctionTransformer(fraudEncoder)),
                          (['Merchant State'], [SimpleImputer(strategy='constant'), FunctionTransformer(np.ravel)
                                                LabelEncoder(), FunctionTransformer(decimalEncoder), OneHotEncoder
                          (['Zip'], [SimpleImputer(strategy='constant'), FunctionTransformer(np.ravel),
                                     FunctionTransformer(decimalEncoder), OneHotEncoder()]),
                          ('Merchant Name', [LabelEncoder(), FunctionTransformer(decimalEncoder), OneHotEncoder()
                          ('Merchant City', [LabelEncoder(), FunctionTransformer(decimalEncoder), OneHotEncoder()
                          ('MCC', [LabelEncoder(), FunctionTransformer(decimalEncoder), OneHotEncoder()]),
                          (['Use Chip'], [SimpleImputer(strategy='constant'), LabelBinarizer()]),
                          (['Errors?'], [SimpleImputer(strategy='constant'), LabelBinarizer()]),
                          (['Year','Month','Day','Time'], [FunctionTransformer(timeEncoder), MinMaxScaler()]),
                          ('Amount', [FunctionTransformer(amtEncoder), MinMaxScaler()])
                         ], input_df=True, df_out=True)
mapper.fit(tdf)

joblib.dump(mapper, open(os.path.join(save_dir,'fitted_mapper.pkl'),'wb'))
```

Figure 3-10 Data transformation – custom mapping

As you can see from Figure 3-11 on page 37, each unique value of the column 'Error?' is converted to a new binary column, with 0 or 1 as the value.

```
mapped_sample = mapper.transform(tdf[:100])
mapped_size = mapped_sample.shape[-1]
mapped_sample.head()
```

Errors?_Bad PIN,Technical Glitch,	Errors? _Bad Zipcode,	Errors?_Bad Zipcode,Insufficient Balance,	Errors?_Bad Zipcode,Technical Glitch,	Errors? _Insufficient Balance,	Errors? _Insufficient Balance,Technical Glitch,	Errors? _Technical Glitch,	Errors? _missing_value	Year_Month_Day_Time	Amount
0	0	0	0	0	0	0	1	0.400008	0.519753
0	0	0	0	0	0	0	1	0.400009	0.387296
0	0	0	0	0	0	0	1	0.400102	0.508274
0	0	0	0	0	0	0	1	0.400147	0.515606
0	0	0	0	0	0	0	1	0.400196	0.493512

Figure 3-11 Data transformation – feature construction

3.3.5 Data enrichment

Data enrichment is the process of enhancing raw data by adding information and context to improve the relevance, value, and informativeness of the data for analysis or ML tasks.

In reality, data could reside in different data sources and in different formats, for example, transaction data can be stored in DB2 on z/OS, whereas the information about each client can be stored on VSAM, and some other information may come from 3rd party or public sources. Integrate relevant information from multiple sources generate a more comprehensive data set, which could improve the value and quality of the data and provide added context and drive more infights. In section 3.4, we discuss how data from different sources can be integrated efficiently and securely.

Data enrichment include different ways to enrich and augment the data. Data from different sources can be integrated together to generate a more comprehensive data set. New data features can be created based on the information from other sources. External information may suggest new ways to process missing values. The data profiling step discussed earlier should be helpful to understand the data from different sources and decide on the data enrichment strategy.

IBM provides tools for data enrichment. Watson Knowledge Catalog and Watson Discovery can help user with data enrichment tasks.

3.3.6 Data validation

Data validation is the process of verifying and validating that the preprocessed data is accurate, consistent, complete, and fit for the intended modeling task before it is used for analysis or model building. Data is the fuel for AI model building, and the quality of the data directly impact the quality and efficiency of the AI model.

As discussed, data validation can be performed in the following dimensions:

- Completeness - This represents the amount of data that is usable or complete. If there is a high percentage of missing values, it may lead to a biased or misleading analysis if the data is not representative of a typical data sample.
- Uniqueness - This accounts for the amount of duplicate data in a data set. For example, when reviewing customer data, you should expect that each customer has a unique customer ID.
- Validity - This dimension measures how much data matches the required format for any business rules. Formatting usually includes metadata, such as valid data types, ranges, patterns, and more.

- Timeliness - This dimension refers to the readiness of the data within an expected time frame. For example, customers expect to receive an order number immediately after they have made a purchase, and that data needs to be generated in real-time.
- Accuracy - This dimension refers to the correctness of the data values based on the agreed upon “source of truth.” Since there can be multiple sources which report on the same metric, it is important to designate a primary data source; other data sources can be used to confirm the accuracy of the primary one. For example, tools can check to see that each data source is trending in the same direction to bolster confidence in data accuracy.
- Consistency - This dimension evaluates data records from two different data sets. As mentioned earlier, multiple sources can be identified to report on a single metric. Using different sources to check for consistent data trends and behavior allows organizations to trust the any actionable insights from their analyses. This logic can also be applied around relationships between data. For example, the number of employees in a department should not exceed the total number of employees in a company.
- Fitness for purpose - Finally, fitness of purpose helps to ensure that the data asset meets a business need. This dimension can be difficult to evaluate, particularly with new, emerging data sets.

3.4 Data integration patterns

Data is an important asset. The IBM Z platform is used for conducting core business transactions across various industries leading to a wealth of organizations’ private data. Unleashing the power of this transactional data, turning it into valuable insights and supporting new business opportunities can bring businesses competitive advantages.

Many factors go into choosing the data integration pattern to satisfy business needs. Companies should start with identifying the business problems at hand, understand the requirements of the use cases, and evaluate different options.

The following questions are some examples of the factors you should consider and evaluate before choosing the data integration pattern:

- What data is needed?
  - What are the data sources, are they from databases or file system?
  - What are the data schemas and data formats?
  - What integration is needed among data sources?
  - What are the purposes of the data processing, is it feeding applications, training AI models, or generating data products?
- Where is the data and where does it need to go?
  - Where does the data need to be processed and eventually be consumed, on-premises, in cloud, or hybrid?
- When is the data needed?
  - Are there real-time requirements?
  - Is the data needed for in-transaction inference in real-time or for batch processing with delay?
  - Is near real-time also accepted?
    - If yes, with how much delay (a few seconds, a few minutes, 1 hour, and so on)?
  - How often does the data need to be refreshed?

- How will the data be made available?
  - Can the data be accessed in place via virtualization, or data copy needs to be maintained in cache or off-premises?
  - What is the granularity and volume, is it transaction level or large volume data dump?
- Why does data need special handling?
  - Is there PII data?
  - Are there regulations to meet?
  - Can the data leave on-premises or certain region?
  - Are there other requirements on latency, performance, security, compliance, auditing, and so on?

Companies can evaluate different options and adopt the right data integration pattern, or a combination of patterns based on the business requirements. We will discuss the different data integration patterns in the next section.

3.4.1 Data exposure pattern: enable modern access to IBM Z data

Mainframes processes 12.6 billion transactions per day, 29 billion ATM transactions annually, and counted on for USD 7.7 trillion in annual credit card.[3] Mainframes usually handle mission-critical business operations and require the highest level of security. These mission critical System of Record (SOR) data resides on mainframe and is hard to access. Traditionally companies have tried to copy data from various sources into central data stores for various business cases such as operational business transactions and analytics. Establishing and maintaining data replication pipelines is expensive, time consuming, and it creates data quality and data latency challenges for consuming applications. Accessing data in place can accelerate transformation and improve its opportunity for success.

Providing modern support for accessing IBM Z data through SQL-based query and through REST APIs could help enterprises to make better usage of the data, whether to support applications on-premises or to support new cloud-native applications. This modern data access simplifies new application development without disrupting data management and recovery processes on IBM Z to maintain data consistency.

IBM Z supports modern access to real-time transactional data in IBM Db2, IMS, and other data sources. Db2 and IMS can be accessed through SQL and REST API by using IBM z/OS Connect EE. Other data sources can be accessed through SQL or REST API by using IBM Data Virtualization Manager for z/OS along with z/OS Connect EE. Data Virtualization Manager for z/OS extends SQL access to data sources other than relational databases as shown in Figure 3-12 on page 40.

[3] The modern mainframe: A banking platform from the future - https://www.ibm.com/downloads/cas/8MGOLOB7

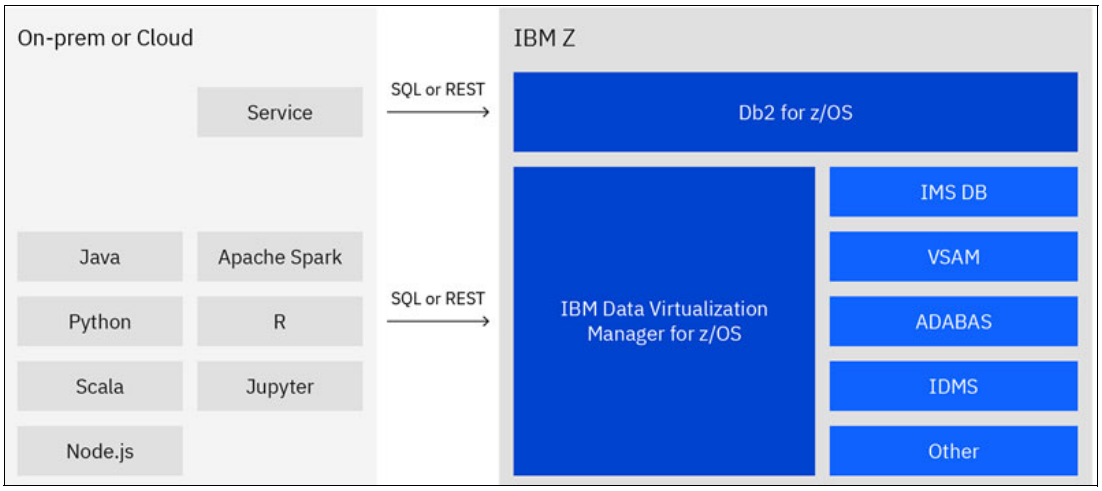


Figure 3-12 Enable modern access to IBM Z data

With Java Database Connectivity (JDBC) support, new cloud applications that consume information from core Db2 and IMS systems can use SQL to detect the underlying data format and contexts from systems of record, which is often required.

You can access data that is stored in Db2 from anywhere by using SQL. Db2 for z/OS also supports native RESTful services to expose SQL and stored procedures as REST APIs when combined with z/OS Connect EE. You can invoke Db2 native RESTful services from z/OS Connect EE by using the z/OS Connect EE REST Client Service Provider.

With z/OS Connect EE, you can consume IBM Z data that is stored in Db2, IMS, or data sources through REST API. Applications anywhere can consume data that is stored on z/OS. It is a common interface for cloud-native applications as shown in Figure 3-13.

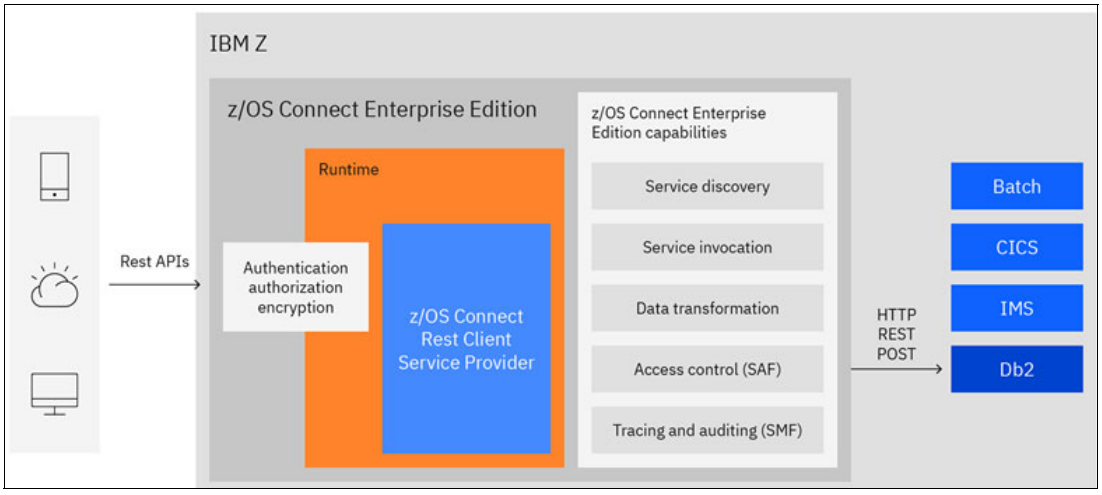


Figure 3-13 Components to support modern access to IBM Z data

Accessing IBM Z data in-place provides several critical business benefits, including the following:

- Reduces the risk of data integrity.
- Reduces the cost that is involved in data movement.
- Increases data quality.
- Preserves the existing data management and recovery processes.

- Allows cloud applications to access the data at its underlying format and context.
- Supports all popular access methods, such as SQL and REST APIs.
- Benefits from higher performance by accessing data that is on IBM Z.
- Avoids the introduction of security intrusion points related to moving the data outside the platform.

By using enable modern access to IBM Z data pattern, you can deliver modern applications, as it facilitates and simplifies access to relational and non-relational IBM Z transactional data and combines that data with off-platform data. The pattern allows access and updates to live IBM Z data through traditional APIs such as SQL and, when combined with IBM z/OS Connect EE, to modern RESTful APIs. It also reduces the cost and delay of moving data to non-IBM Z platforms.

3.4.2 Data virtualization pattern: virtualize IBM Z data

Data is an integral element of digital transformation for enterprises. New services need simplified access to IBM Z data for business operations that require read and updates through APIs. Frequently, IBM Z data also needs to be combined with other data sources.

But as organizations seek to use their data, they encounter challenges that result from diverse data sources, types, structures, environments, and platforms. Those challenges apply equally to data that is stored on IBM Z, which contains most operational data in large organizations. A common concern is that data on IBM Z is difficult to access and transform.

One approach is to move all data into a single data store, such as an operational data store (ODS) or a data lake, which can create more challenges. The complexity of data copy processes results in data latency, poor data quality, increased cost, risks, and security challenges.

Data virtualization is a data integration method that brings together disparate data sources to create virtualized, integrated views of the data without the need to copy and replicate data. Data virtualization also provides a data services layer that abstracts the underlying physical implementation of the individual data sources. Data virtualization helps meet compliance requirements since there are less copies of data, and data is fresh and accurate from source.

The foundation for consuming IBM Z data through data virtualization across data sources is the implementation of the Enable modern access to IBM Z data pattern. That pattern supports access to real-time transactional data in IBM Db2, IMS, and other data sources. You can access Db2 and IMS through SQL, Java Database Connectivity (JDBC), and REST API by using IBM z/OS Connect EE. Data Virtualization Manager for IBM z/OS can provide SQL access to all IBM Z data sources. For through REST API, you can add z/OS Connect EE. For more information about the enable modern access to IBM Z data pattern, see:

https://www.ibm.com/cloud/architecture/architectures/z-enable-modern-access-pattern

The term data virtualization is overloaded. The main adopted use case for Data Virtualization Manager for IBM z/OS is the mapping of traditional IBM Z data sources, such as VSAM, IMS, or Adabas, into relational views for modern access through SQL or API. In contrast, the main use case for data virtualization in IBM Cloud Pak for Data (CP4D) is to gain a single view of disparate data without data movement and to manage data with less complexity and risk of error. For more information about IBM Data Virtualization in CP4D, see:

https://www.ibm.com/products/watson-query

The foundation for accessing data across disparate data sources is the IBM Watson Knowledge Catalog in IBM Cloud Pak for Data. It is more feasible and less costly to maintain

metadata across different data sources instead of constantly moving terabytes of changing data. Watson Knowledge Catalog is a data catalog tool that powers the intelligent, self-service discovery of data structures, models, and more. You can access, curate, categorize, and share data, knowledge assets, and their relationships wherever they are, backed by active metadata and policy management. The cloud-based enterprise metadata repository also activates information for AI, ML, and DL. As shown in Figure 3-14, in Watson Knowledge Catalog, you can discover, govern, and catalog the metadata of IBM Z data that is stored in Data Virtualization Manager for IBM z/OS and Db2 for IBM z/OS, and disparate other data sources.

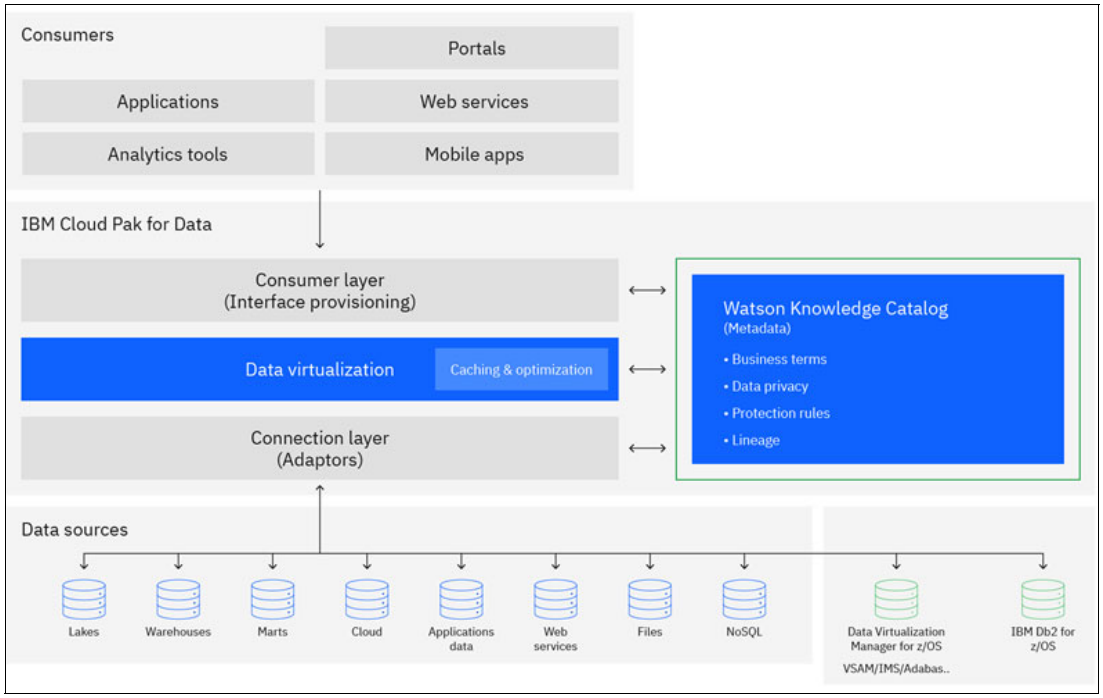


Figure 3-14 Virtualize IBM Z Data

IBM data virtualization is designed as a peer-to-peer computational mesh, which offers a significant advantage over a traditional federation architecture. By using innovations in advanced parallel processing and optimization, the data virtualization engine can rapidly deliver query results from many data sources. Collaborative highly paralleled compute models provide superior query performance compared to federation, up to 430% faster against 100 TB data sets.[4] IBM data virtualization has unmatched scaling of complex queries with joins and aggregates across dozens of live systems. IBM Z data can be accessed through SQL.

Data virtualization can simplify development of consuming applications, including infusing AI into business applications. It also allows those applications to access current and accurate data at its source.

Accessing IBM Z data in place provides several critical business benefits:

- Reduces the risk of impacted data integrity.
- Reduces the cost that is involved in data movement
- Increases data quality.
- Preserves the existing data management and recovery processes.
- Allows cloud applications to access the data at its underlying format and context.

[4] Simplify data access and reduce operational costs while advancing your business insights - https://www.ibm.com/downloads/cas/M76BGEXW

Data virtualization in IBM Cloud Pak for Data is the foundation for rapid ML model development and deployment, infusing AI into business applications.

With a centralized view of data, including IBM Z data, within Watson Knowledge Catalog, you can build, test, and train ML models on the platform of your choice. You can then smoothly deploy AI models to Machine Learning for IBM z/OS to address more complex information needs within business services that run on z/OS as shown in Figure 3-15.

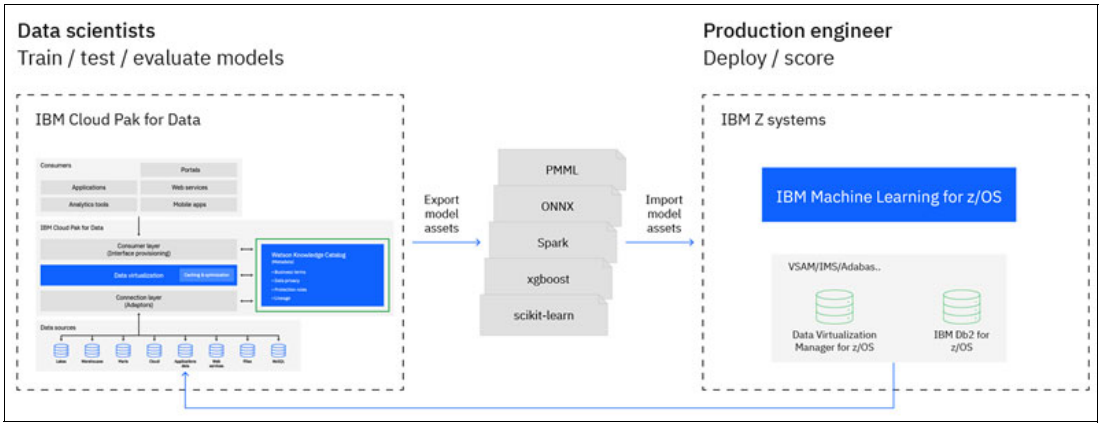


Figure 3-15 AI model building with data virtualization

By using virtualize IBM Z data pattern, you can access data across IBM Z and other data sources, including joining data, without the need to copy and replicate that data. Deliver more current and accurate data with in-place data access to consuming applications, including analytics.

For more information about the virtualize IBM Z data pattern, see:

https://www.ibm.com/cloud/architecture/architectures/z-virtualize-ibm-z-data-pattern

3.4.3 Real-time information sharing pattern: cache IBM Z data

The surge of new digital applications is driving huge growth in data access. Those new applications typically run read-only queries of up-to-date data but are not necessarily associated with generating revenue for the organization. Examples of such applications include mobile banking, retail online browsing, and insurance open enrollment. The characteristics of those applications can make them difficult to plan for and size:

- High, unpredictable volume
- Massive, sharp spikes in activity
- Updates are possible, but are not propagated back to the source (read-only)
- Expensive to maintain
- Complex to implement
- Often compromised data currency
- Prone to instability

Historically, to address those challenges, organizations used several methods to extract data for use on other platforms. It did not matter whether the applications were analytic or simple query-type access, the typical approach was to take the data off platform. Organizations used that approach because website developers who were accessing the data were sometimes unfamiliar with the mainframe and because of concerns that too much read-only activity might conflict with operational applications.

Traditional incremental copy and ETL approaches are unpredictable and can be associated with data latency. By using general-purpose incremental copy and ETL technologies, you can limit efficiency and performance improvement opportunities. Data extraction and incremental copy from IBM Db2 for z/OS can use considerable resources, increase software-related costs, and compete for the same resources that are used for operational processing.

As these applications are often leveraged by end users, the data must be up to date-only moments behind a transactional system. This requirement drives the need for more complex application logic that ensures data currency. On top of the processes to refresh the read-only data store, some organizations use customized code to ensure this consistency. Doing so adds more complexity, instability, and cost to many environments.

Caching support on IBM Z improves application response time by storing copies of data that is on IBM Z. IBM Z offers several products that implement variations of this pattern. The IBM Db2 Analytics Accelerator, IBM Z Digital Integration Hub, IBM Db2 for z/OS Data Gate, IBM Z Table Accelerator, and IBM Data Virtualization for z/OS with Cache Option include an implementation of the cache. Some of the products include a synchronization component to maintain the cache.

The two main differentiators of the cache are as follows:

- Pull versus push maintenance of the cache

  In push maintenance, the cache is always kept in-sync with the source regardless of actual use. Pull maintenance involves a lazy update and invalidation of the cached values based on access.

- Cache data structure

  The data structures of the cache are optimized for the consumption pattern, such as columnar or in memory. The cache itself might also contain derived or precomputed results.

Optimizing application performance by accessing cached IBM Z data can provide several critical business benefits:

- Achieves service level agreements (SLAs).
- Improves performance in scenarios where data is read repetitively and at high frequencies.
- Provides a good compromise between cost and complexity.
- Improves efficiency by avoiding hitting database or other data sources every time for the same request.
- Integrated, lightweight data synchronization.
- Excellent performance and resiliency.
- Lower latency and better data currency.

While caching is commonly used to improve application latency, a highly available and resilient cache can also help applications scale. By using cache IBM Z data pattern you can off load responsibilities from the application's main logic to the caching layer, and free up compute resources to process more incoming work. Read-intensive applications can greatly benefit from implementing a caching approach as shown in Figure 3-16 on page 45.

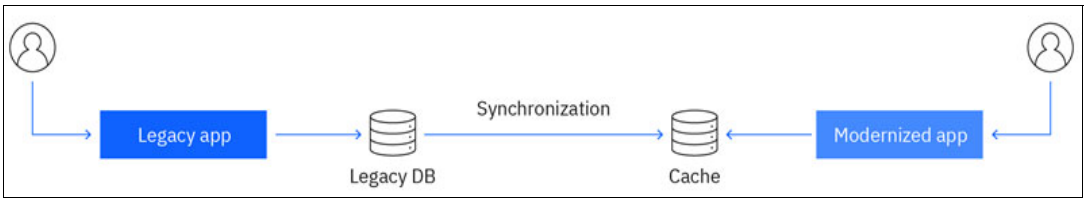


Figure 3-16 Cache IBM Z data

For more information about the cache IBM Z data pattern, see:

https://www.ibm.com/cloud/architecture/architectures/z-cache-ibm-z-data-pattern

3.4.4 Data transformation pattern: transform IBM Z data

The transformation of SOR data by software processes to create a wholly new data set is a specialization of a broader copy-and-access use case. That use case also includes extract, transform, and load (ETL) processes, software replication processes, and virtualization and federation processes. All those broader processes include some transformation capabilities, and it is common to require light transformations during otherwise routine copy-and-access use cases.

This pattern involves heavy set-based transformations that create a data set that is composed by external feeds, derived transformations, and source SOR data sets, such as aggregation or consolidation, and summation. Currency requirements dictate the use of either real-time mechanisms, such as virtualization, near-real-time mechanisms, such as software replication, or periodic batch mechanisms, such as ETL or extract, load, and transform (ELT).

You might need combinations of those three mechanisms if set-based transformations, which are typically the province of ETL products, are required in addition to real-time or near-real-time currency. While this pattern applies to SOR data on any platform, its use with SOR data on IBM z/OS is relevant and valuable because many large enterprises use z/OS as an SOR.

In the simplified depiction presented in Figure 3-17, the transformation processes are indicated by the Data Adapter box. In practice, these transformations can span both processes and time.

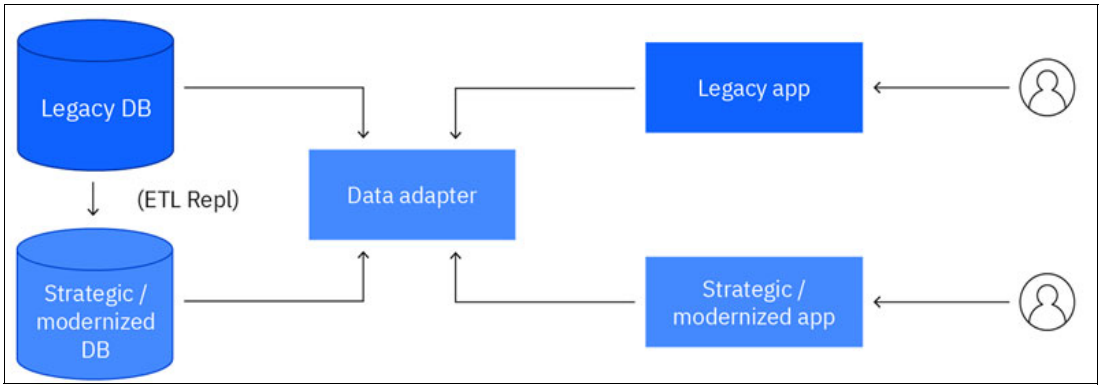


Figure 3-17 Transform IBM Z data

On z/OS, it is typical to have solutions that maintain logically related data across different stacks, such as IBM Db2, IMS, VSAM, or sequential files. For example, transaction data can be stored in DB2, whereas the information about each client can be stored on VSAM. The

ability to view this data through federated queries or from an aggregated copy has value in and of itself. Add the ability to extend the aggregation to derive data (summations, transformations) and to add sources (distributed databases, external feeds), and the value of the original z/OS SOR data grows without disrupting the original workloads.

Beyond data creation, you can use this pattern to create schemas over data. One common use case is to convert normalized data models from the SOR to a de-normalized data model. A de-normalized data model might be one that is used in data warehouse solutions and dimensional data marts that are optimized for statistical and analytical processing. Another common use case is to create user-oriented schemas from what might be a product orientation at the SOR. This use case enables consumption by people who are less familiar with the internal view of the technology or products that underpin the data.

IBM has industry-leading product capabilities in the data transformation space. One such product is IBM DataStage®, which is available in on-premises, on IBM Cloud Pak for Data, and in cloud implementations.

Creating data sets with extra attributes that are derived from production data sets allows for extendibility with minimal disruptions. It also reduces the need to maintain the same data in multiple stores without synchronization.

- No changes are required to the original SOR data sets. All these methods apply transformations downstream from the original data.
- These methods all provide remote access to SOR data in addition to the transformation of that SOR data.
- Downstream data sets can be created or modified without impacting the source SOR data that they are derived from.
- New applications can be created outside of the context of the original SOR data.
- The impact to core SORs from downstream workload requirements is mitigated.

By using transform IBM Z data pattern, you can incrementally build new modernized system-of-record (SOR) data stores by tapping into IBM Z data traffic through a data adapter to transform to the modernized data format. For more information about the transform IBM Z data pattern, see:

https://www.ibm.com/cloud/architecture/architectures/z-transform-data-pattern

3.4.5 Data synchronization pattern: replicate IBM Z data

The replication of system-of-record (SOR) data by software processes to create an alternative, typically remote, copy of that data is a specialization of a broader copy use case. That use case includes extract, transform, and load (ETL) processes, unload and load utilities, and hardware (disk) replication. As opposed to other methods, the specialization that software replication provides is two-fold. Software replication adds both near real-time replication with recovery point objectives (RPOs) near zero and continuous availability with recovery time objectives (RTOs) near zero. While this pattern is applicable to SOR data on any platform, its use with SOR data on IBM z/OS is especially relevant and valuable because many large enterprises use z/OS as an SOR.

Many business drivers exist for replicating SOR data in real time. It is important to note that replication is often deployed as a component of a larger solution. Most customers have “roll your own” infrastructure support, and a subset deploy packaged solutions. These use cases are common:

- The need for mission-critical, workload-level continuous availability at distance to support both planned and unplanned outages.
- The need for a real-time data warehouse to provide accelerated inquiry processing, off loading the SOR.
- The need for advanced analytics processing on real-time data by using various modern analytics platforms such as Hadoop and Apache Spark.
- The need to support various application modernization use cases both in the cloud and on-premises.
- The need to support distributed query use cases such as CQRS, off loading inquiry from the SOR.
- The need to support the aggregation of data from multiple sources into an operational data store (ODS) or a data lake.
- The need to support the off load of the SOR from directly supporting multiple consumers, evolving consumers, or both by replicating into a data lake.

You can implement these drivers by using either homogeneous replication (like models) or heterogeneous replication (unlike models). The diagram in Figure 3-18 is a simplified depiction of a replication setup from a source DBMS to either a target DBMS, a publish/subscribe such as Kafka, or a file system.

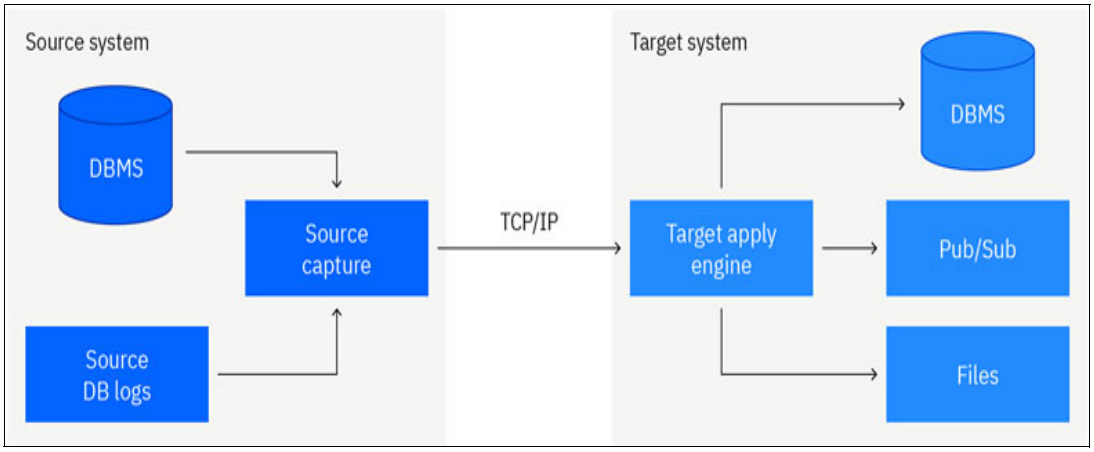


Figure 3-18 Replicate IBM Z data

Continuous availability is usually a homogeneous replication use case. When the SOR is IBM Db2 for z/OS, IMS, or VSAM, the target system is Db2 for z/OS, IMS, or VSAM. Optionally, you use IBM Z GDPS® Continuous Availability, which augments replication with a centralized control plane, intelligent routing, automation, and monitoring. You can also develop these higher-level constructs over replication by using alternative, often home-grown, methods.

The other business drivers are heterogeneous in nature. You can usually deploy the target systems and DBMS or file systems for those drivers in Linux on IBM Z, z/OS Container Extensions, or on IBM Z as an alternative to running a few components on distributed systems. An “all on IBM Z” approach has the advantage of providing better high availability characteristics, reliability, security, and management. Performance is also better, as fewer availability zones and network devices need to be traversed.

IBM has a family of replication products that are referred to as IBM InfoSphere® Data Replication and IBM Data Replication. This family includes multiple technologies, including QRep, CDC, and Classic products.

For more information, see the InfoSphere Data Replication IBM documentation:

https://www.ibm.com/docs/en/idr/11.4.0

The use of software replication to create one or more copies of SOR data provides several key advantages:

- Software replication is a near real-time synchronization method. Average latency for online processes, which is sometimes referred to as transactional, is usually measured in low seconds if not sub seconds. The average latencies for batch processes are also low but are higher than online processes. Batch latencies are usually measured in minutes. Those latencies meet many companies' SLAs for RPO. Other techniques are either periodic pulls of all data, such as once a day, or replication to auxiliary storage devices that must be varied online for use.
- Software replication captures changes to source data as they occur and replicates those changes to target agents that replay those changes directly through the DBMS or file system stack. This process enables the RPO to become the RTO, which means that you can achieve RTOs of low seconds to sub seconds from any change.
- Software replication can run at any granularity and tends to be focused on the sets of data for an application or a workload. You can implement custom configurations and processing at the workload level. In contrast, other solutions such as disk replication tend to be site-level in nature and constrained by the proper usage at the disk volume level.
- Software replication processes data at transaction boundaries and can maintain target copies to be transactionally consistent with the source. This advantage is not possible with alternatives such as ETL or disk replication.
- Costs are optimized by off loading queries and analytics processing to the target systems. This behavior reduces the strain on the SOR but still achieves CPU-intensive goals.
- You can handle large volumes in terms of throughput of processing and still maintain low latency.
- The impact of spiky and unpredictable workloads to core SORs is mitigated.

When you implement software replication from SOR data, you need to consider a few factors. First, real-time replication requires supplemental logging. Replicating products read change records from DBMS recovery logs. As their name implies, those logs are designed for use by DBMS recovery processes and contain content that is sometimes tailored for those purposes. Replication, unlike recovery, requires more attributes and transactional semantics, such as commits, rollbacks, and operations that are identified by their transaction ID, before images for updates, open and close events, and load utility events.

Augmenting recovery logs to add in those attributes increases the volume of the changed data to the logs, which often affects log switch procedures, and the impact to the source application response time. You can mitigate some of the impact to the source application response time by using good buffering and off loading designs. In addition, log retention periods are often increased for replication use cases to allow for earlier restarts. Adding supplemental logging might require more capacity and can affect log switch times.

Real-time replication can be CPU intensive. The reading of logs, merging of change records, transactional buffering, refreshes (select *), transmission, and parallel replay all affect CPU. Software replication is often compared unfavorably to disk replication, its older sibling, as disk replication often is done at the control unit, off-platform. Adding replication might require more capacity to achieve the best performance as measured by latency.

Real-time replication provides a tight synchronization between source and target data. Effectively, the target copy is an extension of the source copy and must be considered for any

source-side change management, batch processing, and utility processing such as reorgs. You might be tempted to treat the copies that replication maintains as being loosely coupled. However, in most use cases, many data-centric source procedures must be extended to consider the effect on the copy.

By using replicate IBM Z data pattern, you can replicate data in real time by capturing change log activity to drive changes in the target. Enable newer applications to access a broader range of data stores and access methods through replication. For more information about the replicate IBM Z data pattern, see:

https://www.ibm.com/cloud/architecture/architectures/z-replicate-data-pattern

4

Model building for IBM Z

Model building is a critical step within the AI lifecycle. The decisions you make around model training will make a significant impact on the model's performance when it is put into production. We have previously discussed the art of data engineering, which integrates very closely with model training. The model we are building will only be as good as the quality of the data that is used to train it.

4.1 Model training frameworks

When building an AI model, it is extremely helpful to make use of a training framework. A training framework will provide the tools necessary to build, train, and validate models in an efficient way. These efficiencies will give you the ability to build models in an agile fashion.

4.1.1 Model training frameworks and technologies on IBM Z

There is a wide variety of common model training frameworks that are available today. There is a vast range of open-source frameworks, but also IBM provided solutions that reduce the complexity for a developer or data scientist when navigating through the full AI lifecycle.

The IBM provided solutions that reduce complexity and enable the full AI lifecycle include the following IBM products:

- Machine Learning for IBM z/OS

  Machine Learning for IBM z/OS (MLz) not only provides the Jupyter Notebook functionality to train models directly within its user interface, but also has the service capability that users can leverage to run, manage, and automate the model training process on z/OS Spark. Here you can build out custom training pipelines that interface with your data and save them directly within your environment.

  For details on the procedure for developing a model in the integrated Notebook Editor in MLz, see:

  https://www.ibm.com/docs/en/wml-for-zos/enterprise/3.1.0?topic=wmlz-developing-models-in-notebook-editor

  For more information regarding submitting application to Spark for model training with MLz, see:

  https://www.ibm.com/docs/en/wml-for-zos/core/3.1.0?topic=apis-submitting-application-spark-model-training

- IBM Cloud Pak for Data

  IBM Cloud Pak for Data (CP4D) is a modular set of integrated software components for data analysis, organization, and management. As part of the vast feature set of CP4D, it includes the ability to train AI models. You can use Watson Studio and Machine Learning to harness the ability to build custom training pipelines with tools like Jupyter notebook within CP4D. In addition, the AutoAI feature is an excellent tool to rapidly train and tune a wide range of potential models. CP4D, including the AutoAI functionality, are available on as a software license, as a service, and on prem with IBM Z.

  When using IBM Cloud Pak for Data on IBM Z, you experience the added benefit of the hardware acceleration. It leverages the IBM Z Integrated Accelerator for AI for Snap ML and TensorFlow models.

- Python AI Toolkit for IBM z/OS

  The vast majority of common AI, machine learning, and analytics interfaces are open source and package security requires currency. Python AI Toolkit for IBM z/OS consists of up-to-date validated and secure open source packages optimized for z/OS, delivered through a secure channel in a self-service method. Python AI Toolkit for IBM z/OS provides several different training frameworks and technologies. Some available packages include Scikit-learn, XGBoost, Jupyter notebook, numpy, pandas, matplotlib, and many more.

More information on Python AI Toolkit for IBM z/OS can be found within the product announcement letter:

https://www.ibm.com/docs/en/announcements/python-ai-toolkit-zos-110?region=US

- IBM z/OS Platform for Apache Spark

  Apache Spark is an open source, distributed processing system used for big data workloads. Spark is a part of the Apache suite which is one of the largest open source foundations. Spark provides the ability support ML workloads by automating data processing for real-time as well as batch use cases.

  More information on IBM z/OS Platform for Apache Spark can be found within the product announcement letter:

  https://www.ibm.com/docs/en/announcements/z-platform-apache-spark?region=US

- SnapML

  Some classical ML model types, such as gradient boosting machine, random forest, and extra trees, can take advantage of the IBM Z Integrated Accelerator for AI by using the Snap ML format. Snap ML is an ML framework that is developed by IBM Research Zurich to accelerate the training and inferencing of traditional ML models by efficiently using computing resources through parallelism while maintaining high model accuracy. For development purposes, documentation for the Snap ML format can be found at Snap machine learning:

  https://snapml.readthedocs.io/en/latest/

  Snap ML was designed to deal with the challenges that enterprises face when training or retraining models on vast quantities of data, such as long training times and massive resource utilization. Additionally, Snap ML is seamless for data scientists to use if they are used to creating models in the scikit-learn library, which can be installed by using a simple pip command. Some scikit-learn models after an intermediate conversion to a PMML format can be converted to the Snap ML format to realize greatly reduced inferencing latency, even when running on CPU.

Additionally, there are the following open source frameworks:

- TensorFlow

  TensorFlow, originated by Google Brain team, is one of the most popular model training frameworks used by data scientist. TensorFlow is an open source framework that supports deep learning.

  When running TensorFlow on IBM Z it also takes advantage of the IBM Z Integrated Accelerator for AI transparently. You can find the latest TensorFlow container available in the IBM Z and LinuxONE Container Registry (ICR) to experience accelerated inferencing.

  When training is complete, TensorFlow models can be converted into ONNX format, compiled with the IBM Z Deep Learning Complier (DLC), and deployed with MLz. MLz eases the user experience by compiling the deep learning model under the covers.

- Scikit-learn

  Scikit-learn is another commonly used model training framework. Scikit-learn is an open source framework that supports machine learning, as opposed to deep learning. More specifically, scikit-learn provides functionality for several different classification, regression, clustering, dimensionality reduction, model selection, and preprocessing techniques. It leverages and is built on other widely known open source python packages such as NumPy, SciPy, and matplotlib.

  Scikit-learn can be used to train on or off IBM Z. When training on z/OS, you can acquire this framework through the Python AI Toolkit for IBM z/OS as a standalone, or as part of the MLz training platform.

After the machine learning model has been built, it can be converted to PMML format, then dependent on the model type it can utilize the SnapML runtime and take advantage of the IBM Z Integrated Accelerator for AI.

- PyTorch

  PyTorch is a machine learning library that was originally developed by Meta. It is an open source framework that supports deep learning development for use cases like natural language and image processing.

  You can use the IBM Z and LinuxONE Container Registry (ICR) to pull the latest container with PyTorch support.

  https://ibm.github.io/ibm-z-oss-hub/main/main.html

- Jupyter Notebooks

  Jupyter notebooks are a foundational tool when running various software languages such as Python, Scala, Java, and R.

  It is open source software that provides a web based application that “allows you to create and share documents that contain live code, equations, visualizations, and narrative text.”

  Jupyter notebook allows you to run the various open source training frameworks previously discussed. Also, it integrates directly into MLz for model training. There are several options when using Jupyter notebooks on IBM Z. The standalone python package can be acquired from the Python AI Toolkit for IBM z/OS and a container with Jupyter can be found on ICR.

4.2 Model formats

After training is complete and the model was built successfully, we need to convert it into a usable format that can be leveraged for deployment. Model file formats allow us to perform inferencing with our developed model.

Popular model formats in the industry include the following:

- MLz model

  Training within MLz provides the ability to save the MLz model directly to the MLz repository. When exporting the MLz model, we can make use of several different libraries. MLz support a diverse list of models such as Spark, Scikit-learn, XGBoost, PMML, ARIMA, Seasonal ARIMA, ONNX, Watson Core Time Series.

  For the complete list of supported model formats, see:

  https://www.ibm.com/docs/en/wml-for-zos/enterprise/3.1.0?topic=wmlz-importing-models-from-file

- ONNX

  Open Neural Network Exchange, or ONNX, is an open format built to represent machine learning and deep learning models. It provides a common file format that allows developers and data scientists to train models using different frameworks, then convert the model into the ONNX format.

- PMML

  Predictive Model Markup Language, or PMML, is another standard and open model format. It is XML-based and similar to ONNX, provides a way to represent models that have been trained using different frameworks.

4.3 Training strategy

Having a sound training strategy is important because there's a wide range of available architectures that can be implemented, whether it is on-prem, public cloud, private cloud, or some variation of each. Although AI projects can be solved in many ways, there are some solutions that are more optimal based on the use case.

4.3.1 Train anywhere and deploy on IBM Z

Businesses across the globe are increasingly relying on cloud computing to modernize their current operations. Majority of these businesses across the industries like Banking, Insurance, Health care, Retail, Government etc., run their core transactional and batch workloads on IBM Z as they provide unmatched qualities of service like security, reliability, and availability. To retain the core strengths and attributes of IBM Z platform and to leverage extensive cloud services, security and regulatory compliance programs of IBM Cloud, most scenarios require a hybrid cloud approach.

Cloud offers a great deal of flexibility and scalability, enables effective collaboration, and supports flexible pay-as-you-go pricing model. Across the globe companies are increasingly relying on cloud computing to operate their business. Data scientists can access the various tools and training platforms, utilize DevOps pipelines, and access rich sets of data and applications available in cloud to develop, train, and test the AI models.

Companies need to have intelligence built into the transaction processing logic to handle instant payments, increasingly stricter regulations that require real time Anti-Money Laundering (AML), and the dramatic increase in online transactions, along with the increased amount of fraud. Implementing real-time AI in enterprise workloads without impacting service-level agreements (SLAs) is now more critical than ever. Even for clearing and settlement use cases that are processed in batch workloads the volume of transactions is ever increasing and the time windows are shrinking as the industry is moving to same day settlements. Throughput and latency have become critical for both real-time and batch processing.

Developments such as the IBM Telum processor chip with the IBM Z Integrated Accelerator for AI are positioning IBM as the premier platform provider for inference workloads. Collocating the data preprocessing logic and model inference with core transaction processing logic on IBM Z allows clients to integrate real-time inference into core business transactions and score all transactions in real-time without impacting the mission critical SLAs.

As illustrated in Figure 4-1, enterprise can start with organizing data from different application and sources, and then build and train a model on any public cloud or on-premises infrastructure by using a preferred model framework, finally the model can be deployed on IBM Z to take advantage of the innovative hardware optimizations to infuse AI into every transaction.

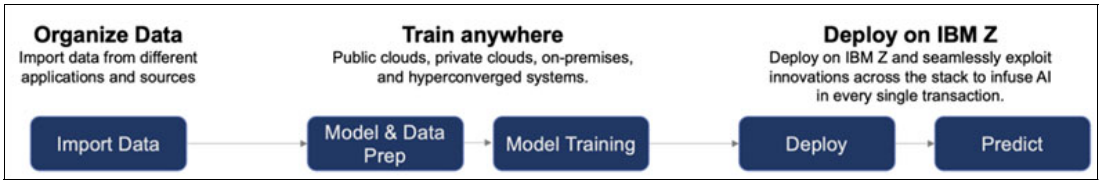


Figure 4-1 Train anywhere, deploy on IBM Z

4.3.2 Set up secure environment in IBM Cloud

Although cloud provides many benefits, security is the main concern when using cloud computing services. According to Cost of a Data Breach Report 2023[1], the average cost of a data breach reached an all-time high in 2023 of USD 4.45 million. This represents a 2.3% increase from the 2022 cost of USD 4.35 million. Taking a long-term view, the average cost has increased 15.3% from USD 3.86 million in the 2020 report. Cost of a data breach could be as high as 5.90 million for financial industry in 2023. Thus, setting up a secure environment in the cloud is critical.

Financial institutions need to comply with cyber security standards, laws, and regulations to maintain a strong security posture and to prevent data breaches. The global regulatory, standards and compliance landscape (see Figure 4-2 is complex and continuously evolving. There are variety of security regulations and international, federal, and regional laws. Financial institutions typically must comply with more than one set of requirements, and it is easy to get lost while implementing relevant IT standards, regulations, and local laws. Financial institutions also need to constantly adjust their security controls and processes to frequent cyber security landscape changes. It is time consuming and challenging to meet the compliance requirements and keep up with the changes.

Figure 4-2 Major global regulatory bodies

There are many different factors to consider when designing a secure environment in cloud, including network security, identity and access control, application security, and data security. It requires deep knowledge in each area to design the infrastructure and make sure that there are no security holes in the design. Once implemented and deployed, the environment needs to be continuously monitored to make sure that they maintain a strong security posture.

IBM Cloud is designed for regulatory workloads and works with an ecosystem of regulated clients and ISVs to continuously improve the compliance posture. IBM Cloud has many features to support Hybrid Cloud deployments and is uniquely positioned to support enterprise workloads including regulated workloads with its support for heterogeneous compute architectures like x86, Z and Power. IBM Cloud also benefits immensely from IBM Z platform centric enterprise cloud services offered in its catalog.

As shown in Figure 4-3 on page 57, IBM Cloud for Financial Services® is part of IBM Cloud. It is comprised of IBM Cloud services and independent software vendor applications that comply with the IBM Cloud Framework for Financial Services. IBM Cloud Framework for Financial Services is designed to build trust and enable a transparent public cloud ecosystem of ISVs and IBM Cloud services with the features for security, compliance, and resiliency features that financial institutions require. The Framework utilizes services in IBM Cloud to

[1] https://www.ibm.com/security/data-breach?_ga=2.268429834.560404878.1653059563-954325317.1653059563

create secure and compliant environment, (for example, IBM Hyper Protect Crypto Services (HPCS), IBM DevOps toolchain, IBM Security® and Compliance Center (SCC)), includes reference architectures and best practices. The Security and Compliance Center has a set of profiles with pre-defined controls and polices that will run, monitor and audit the services. The policies and controls are determined by IBM Cloud Regulatory Council of members who are from risk and compliance of leading financial industry companies and Promontory® Financial Group, an IBM subsidiary, to ensure that it is current with new and updated regulations. The ISVs and IBM Cloud services are being validated by the controls defined by the regulatory council members. Once validated they become Financial Services validated.

The IBM Cloud Framework for Financial Services defines reference architectures, which can be deployed via infrastructure-as-code DevOps toolchain and used as a basis for meeting the security and regulatory requirements. Sensitive workloads (such as, AI model training jobs) can then be deployed into the environment. The reference architectures defined by the IBM Cloud Framework for Financial Services include the following:

- VPC reference architecture for IBM Cloud for Financial Services:

  https://cloud.ibm.com/docs/framework-financial-services?topic=framework-financial-services-vpc-architecture-about

- IBM Cloud Satellite® reference architecture for IBM Cloud for Financial Services:

  https://cloud.ibm.com/docs/framework-financial-services?topic=framework-financial-services-satellite-architecture-about

- IBM Cloud for VMware Regulated Workloads architecture:

  https://cloud.ibm.com/docs/framework-financial-services?topic=framework-financial-services-vmware-overview

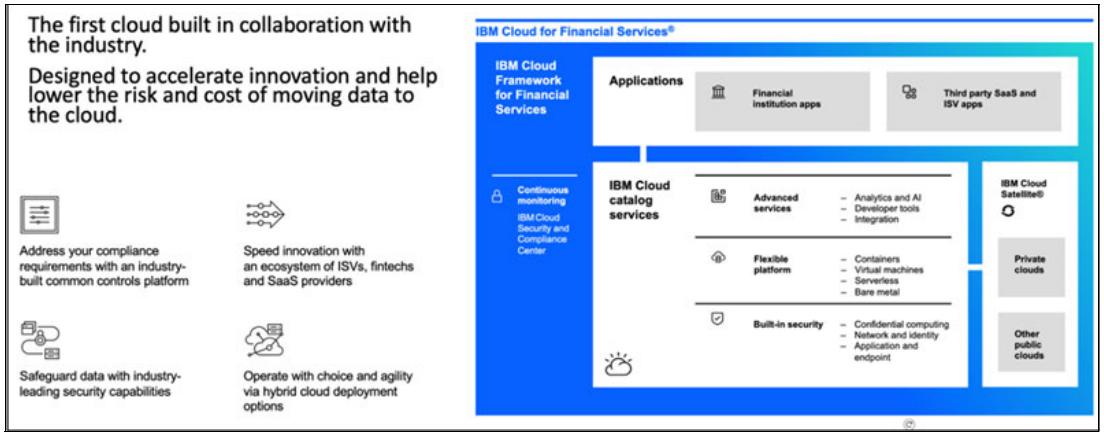


Figure 4-3 IBM Cloud for Financial Services

For more information about IBM Cloud for Financial Services, see:

https://www.ibm.com/cloud/financial-services

4.3.3 Model development in hybrid cloud

Data is needed for the development of an AI model. Depending on the business problem, data from various sources need to be organized and made available for data scientists to access. Section 3.4, “Data integration patterns” on page 38 discusses various data integration patterns that enterprises can choose based on the requirements of the use cases.

Data scientists can develop, train, and test the AI models using his preferred framework anywhere, including IBM Z or in private or public clouds. However, as we discussed in the last subsection, since data could be sensitive and models are valuable assets of the enterprise, the training environment needs to be secure and compliant.

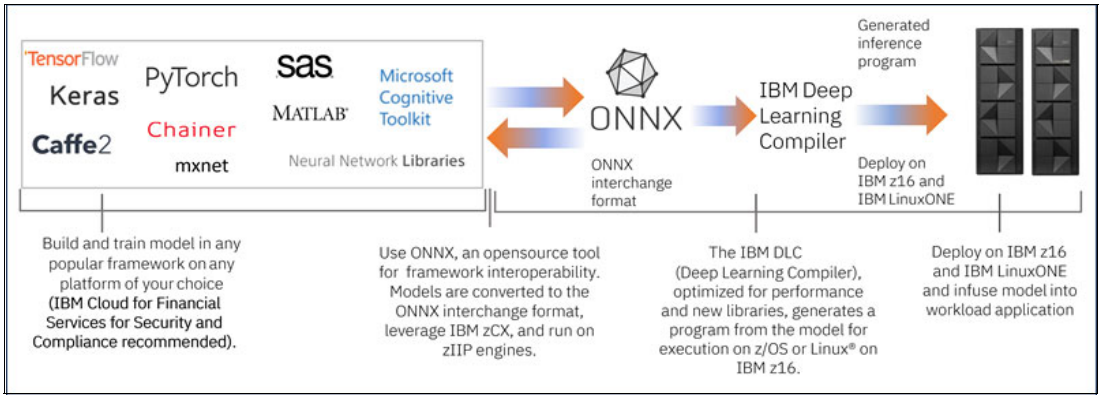


Figure 4-4 Deep learning model development and deployment process

Depending on the business problem at hand, data scientist could build ML models or DL models to solve the problem. Figure 4-4 shows typical DL model development and deployment process. As you can see from the figure, the choice of model development environment could be independent of the model deployment environment. Data scientist can choose to build and train the AI model in any popular framework on any platform of their choice, as long as it meets the security and compliance requirements. Once model is trained and tested, data scientist can convert it to a format that is easier for framework interoperability, for example, Open Neural Network Exchange (ONNX) format. ONNX model can then be managed in secure model repository, and securely transferred to IBM Z for testing and deployment, which is managed by the DevOps pipeline over Direct Link connecting IBM Z with IBM Cloud. ONNX model can be imported into MLz and compiled by Z Deep Learning Compiler (zDLC) under the cover, or it can be compiled manually within a container image of the zDLC, to take advantage of the Integrated Accelerator for AI on IBM Z.

ONNX is not the only format that can be deployed on IBM Z. Figure 4-5 on page 59 shows several supported AI frameworks and libraries that can be used to import many developed models into various runtime environments on IBM Z, and the hardware exploitation that the model can benefit from during inferencing. Many popular model types from several frameworks can be imported into the IBM Z platform seamlessly into their native runtime environment, which typically refers to an Anaconda, Python, or Spark environment in Linux on IBM Z or zCX.

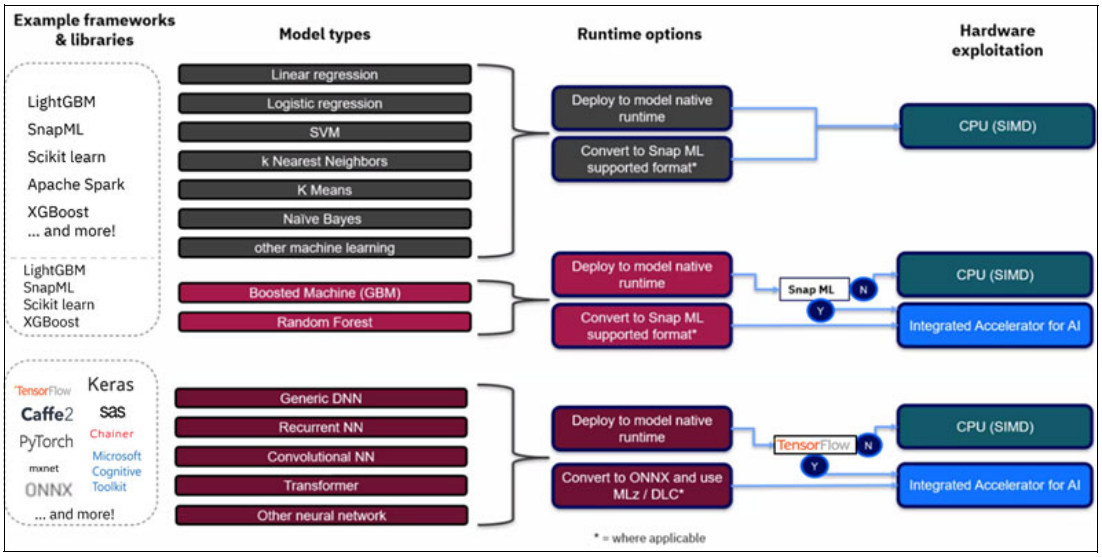


Figure 4-5 Model optimizations on IBM Z

4.3.4 AI reference architecture in hybrid cloud

Enterprise data is highly sensitive and needs to be protected, and AI workloads also need to be deployed in a cloud environment that meets regulatory compliance, security, and resiliency requirements. As we discussed in the previous section, IBM Cloud for Financial Services is designed to help clients mitigate risk and accelerate cloud adoption for even their most sensitive workloads.

AI model building is an iterative process. It normally starts with understanding the business problem at hand, and iterate through understanding and preparation of data, training and testing the models, model deployment, model inference, and model monitoring.

The IBM Cloud Framework for Financial Services provides three reference architectures that can be used as a basis for meeting the security and regulatory requirements that are defined in the framework. Figure 4-6 on page 60 shows a sample AI reference architecture in VPC for a hybrid cloud environment.

For a more detailed overview of the IBM Cloud Framework for Financial Services reference architectures, see:

https://cloud.ibm.com/docs/framework-financial-services?topic=framework-financial-services-reference-architecture-overview

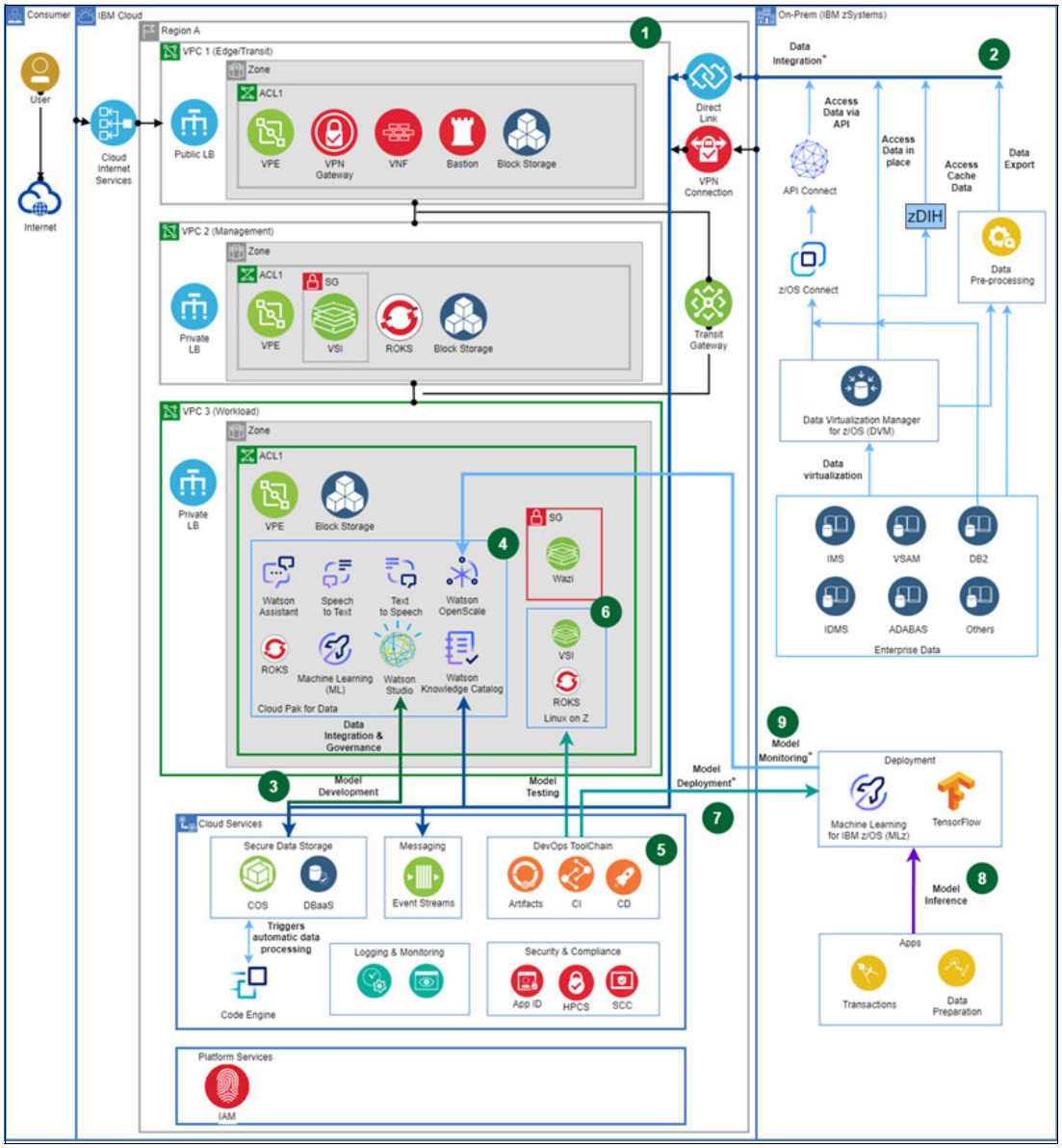


Figure 4-6 Sample AI reference architecture in VPC for a hybrid cloud environment

The following steps outline how to set up an AI reference architecture in VPC for a hybrid cloud environment across IBM Cloud and on-premises:

1. Set up the infrastructure by creating DevOps toolchain for “Deploy infrastructure as Code for the IBM Cloud for Financial Services”, which provides a secure environment for model training jobs. For more information, refer to reference architecture for IBM Cloud for Financial Services.
2. Make the enterprise data on-prem accessible in cloud. There are different ways to expose the data on-prem. Please refer to the data integration patterns discussed in “Data integration patterns” on page 38.
3. Understand, prepare, and manage the governance of the data with tools in Cloud Pak for Data, for example, organizing and managing data with Watson Knowledge Catalog.

4. Training jobs can be run in the workload VPC. Various tools in IBM Cloud Pak for Data can be used. Watson Studio can be used as the model development environment. Different machine learning platforms and tools can be used to train the model (for example, TensorFlow, PyTorch, SnapML, and so on). Model can be tested in Machine Learning. For conversational AI, tools like Watson Assistant, Speech-to-Text, and Text-to-Speech can be used.
5. IBM DevOps Toolchains can be used to manage the model source codes and tested models.
6. Linux on IBM Z mainframe or IBM Wazi as a Service can be used to test the AI models and AI applications before pushing them to on-premises mainframe.
7. Tested AI models are deployed on Machine Learning for IBM z/OS.
8. AI model inference is invoked to gain insights from the data.
9. AI models are continuously monitored via IBM Watson OpenScale, which is an enterprise-grade environment for AI applications that provides enterprise visibility into how your AI is built and used, and delivers return on investment. Its open platform enables businesses to operate and automate AI at scale with transparent, explainable outcomes that are free from harmful bias and drift.

4.3.5 Training on-premises with IBM Z

A significant amount of enterprise data resides directly on IBM Z. Due to this sensitivity of the data; it may be desirable to keep data on prem while performing training.

Training with MLz

As previously discussed, popular model training frameworks allow the flexibility to run within various environments. On IBM z/OS, you can build models using MLz UI, leveraging the Jupyter notebook interface. MLz provides an enterprise ready platform that eases the deployment of the model when training is complete.

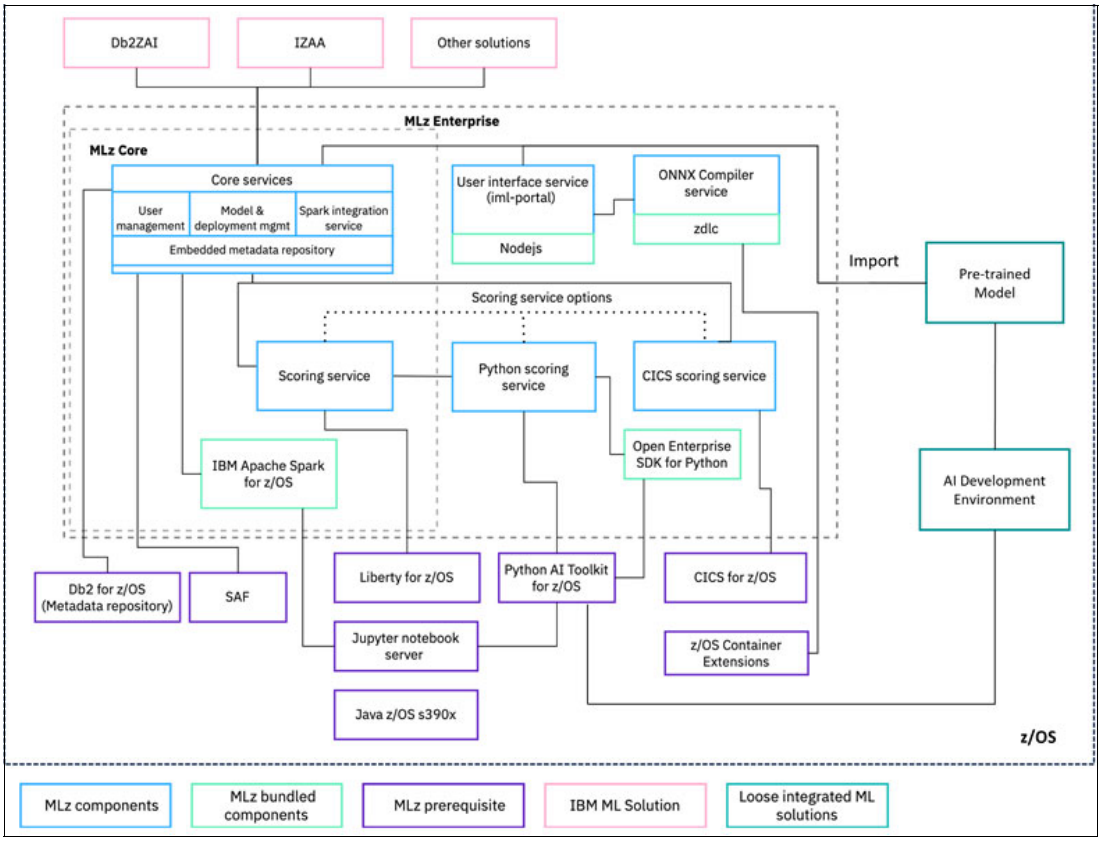


Figure 4-7 MLz architecture

The diagram shown in Figure 4-7 on page 62 visualizes the architecture of MLz with an example showing how to leverage a custom ML solution on z/OS to train a model and import it into MLz.

Training with Python AI Toolkit for IBM z/OS

Additionally, on IBM z/OS you can utilize the Python AI Toolkit for IBM z/OS, which provides some of the most popular Python packages for model training, allowing for a DIY development solution. Some available packages include Scikit-learn, XGBoost, Jupyter notebook, NumPy, pandas, matplotlib, and many more.

All the latest package included within the Python AI Toolkit for IBM z/OS can be found on the product page. For more detailed guidance see the following IBM Redbooks Solution Guide, *Securely Leverage Open-Source Software with Python AI Toolkit for IBM z/OS*, REDP-5709.

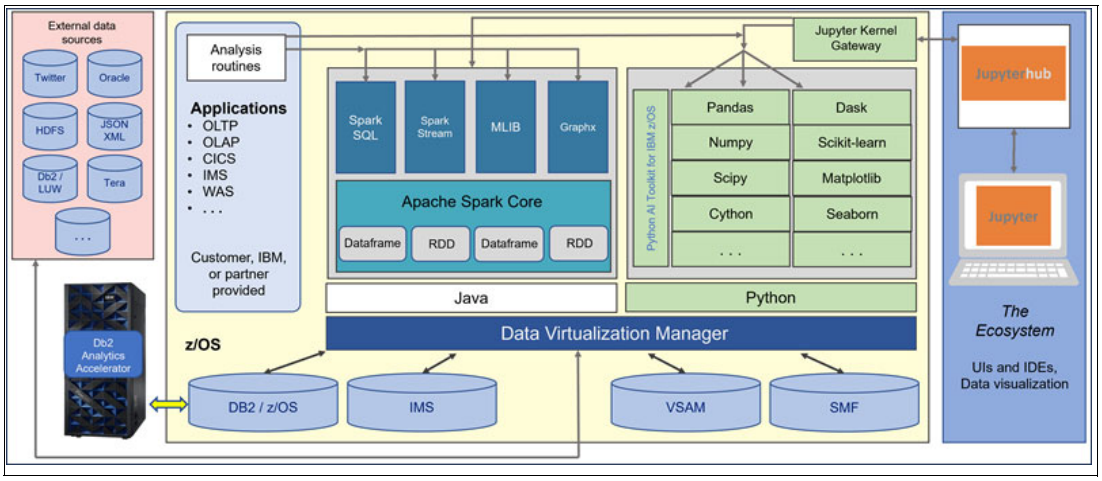


Figure 4-8 Journey to Open Data Analytics on IBM Z

Illustrated in Figure 4-8 on page 63, you can see how to use Jupyter within AI development space to build models with Python AI Toolkit for IBM z/OS. For a more complete environment for AI and open data analytics applications, you can make use of IBM z/OS Platform for Apache Spark and IBM Data Virtualization Manager for z/OS.

IBM Z Platform for Apache Spark allows multiple, disconnected data sources to be virtually integrated within a single application. Apache Spark is the enterprise engine for processing unified data and scaling integrations with business applications.[2]

IBM Data Virtualization Manager for z/OS provides virtual, integrated views of data residing on IBM Z. It enables users and applications read/write access to IBM Z data in place, without having to move, replicate or transform the data.[2]

Training with containers on Linux on IBM Z

You can run popular model training frameworks on IBM zCX or Linux on IBM Z environments, leveraging containers from the IBM Z and LinuxONE Container Registry. These training frameworks also can run within x86 environments, such as Windows, Mac, and Linux.

Some available containers include TensorFlow, PyTorch, and many more. The complete list of the latest containers can be found on the IBM Z and LinuxONE Container Registry website,

https://ibm.github.io/ibm-z-oss-hub/main/main.html

[2] https://www.ibm.com/support/z-content-solutions/journey-to-open-data-analytics/

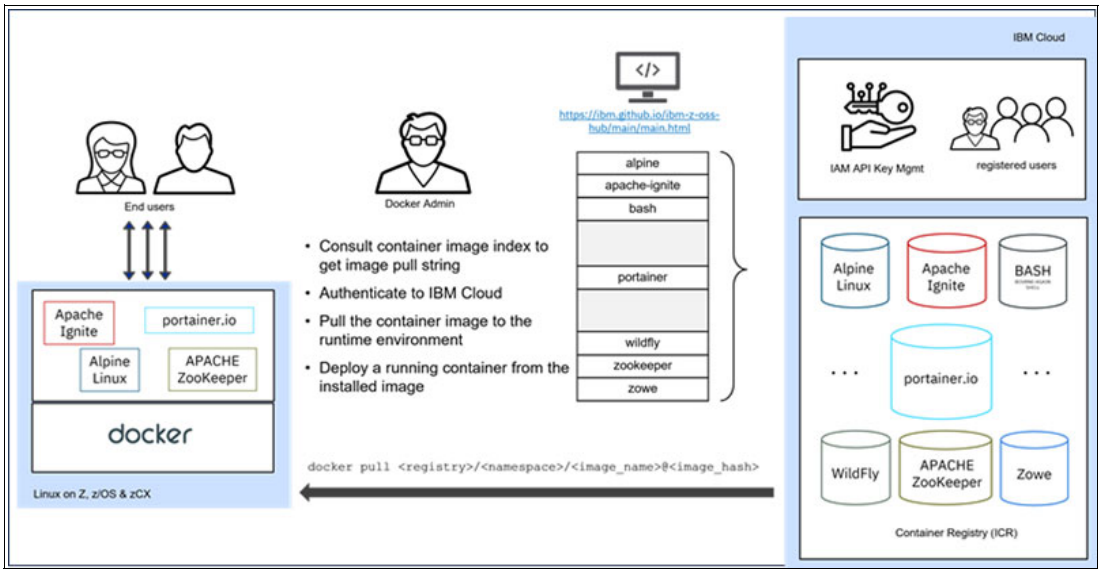


Figure 4-9 IBM Z and LinuxONE Container Repository

Figure 4-9 shows the high-level architecture of an development environment that makes use of ICR. To perform model training within a Linux on Z, z/OS, or zCX environment you can pull the docker files directly from ICR.

4.4 Best practices for model training

Each decision you make when performing model building is directly dependent on the problem that is being solved. Although each problem is different, there's a set of best practices that can be applied.

4.4.1 Training strategy considerations

When building your training strategy there are several variables to consider. It is key to identify where your data currently resides. The ability to access the data is critical to building a model.

Where is the data?

Model training can be a resource intensive process. The size of the data set you're planning to train with will directly impact the time it takes to train your model, in addition to other variables. The quality of the data is always more important than the quantity. The model type and it is corresponding hyper parameters can also significantly impact training times. Finally, the underlying hardware infrastructure and resources available will also determine the speed at which training can be executed.

Model update frequency

Every use case is different, and some use cases require the need to frequently train the model with updated data. For example, if you need your model to make decisions based on data that is generated every hour or day, then you will need to build a new model and deploy it rapidly. The training pipeline considerations may look much different than a use case that only needs historical data from the previous year. Later, we will discuss how MLOps is tightly integrated with the model training process.

4.4.2 Considerations for choosing a framework

When choosing a framework to train your AI models, there are many aspects that come into consideration. It is important to note that there is no perfect one fits all answer, but instead general techniques that can be applied when experimenting.

Deep learning vs. machine learning

Different model types have several strengths and weaknesses depending on the use case you are trying to solve. It is important to consider the strengths of each model in order to find the best fit and maximize the performance.

First, it is important to understand whether the use case leans more towards traditional machine learning or deep learning. If the use case is image or natural language processing related, it is likely you will leverage a deep learning model. For deep learning, you can make use of TensorFlow and PyTorch. If the use case is outside of this space and caters towards traditional machine learning models, you can make use of frameworks like Scikit-learn, Snap ML, XGBoost, or others.

Open source vs. enterprise platform

As we covered earlier in this chapter, there are several different training frameworks at your disposal. Standalone open source frameworks such as TF, scikit-learn, and others allow you to build out your training infrastructure in a sort of DIY approach. The advantage of using MLz is that it provides an end-to-end enterprise platform that allows for seamless integration when moving towards deployment. The important thing to know is that there are many options and the flexibility of MLz supports this.

Cloud vs. on-premises

Whether you choose to train on prem or in the cloud, there are many frameworks and technologies available. In fact, most of the same technologies are available native on IBM Z that are available on IBM Cloud as well as other cloud providers. If you choose to train natively on z/OS, MLz is the full end-to-end enterprise platform of choice. If you're interested in developing your own model training infrastructure, you can leverage the wide range of open source python packages available in the Python AI Toolkit for IBM z/OS.

4.4.3 Best practices on saving models for deployment on IBM Z

Once training is complete, it is now time to save the model so that it can be used for deployment. When training with MLz, you can easily save your model directly within your environment. As previously mentioned, you can reference how to save the model using the integrated Notebook Editor in MLz. You can use the MLz UI to import a previously exported MLz model, as shown in Figure 4-10 on page 66.

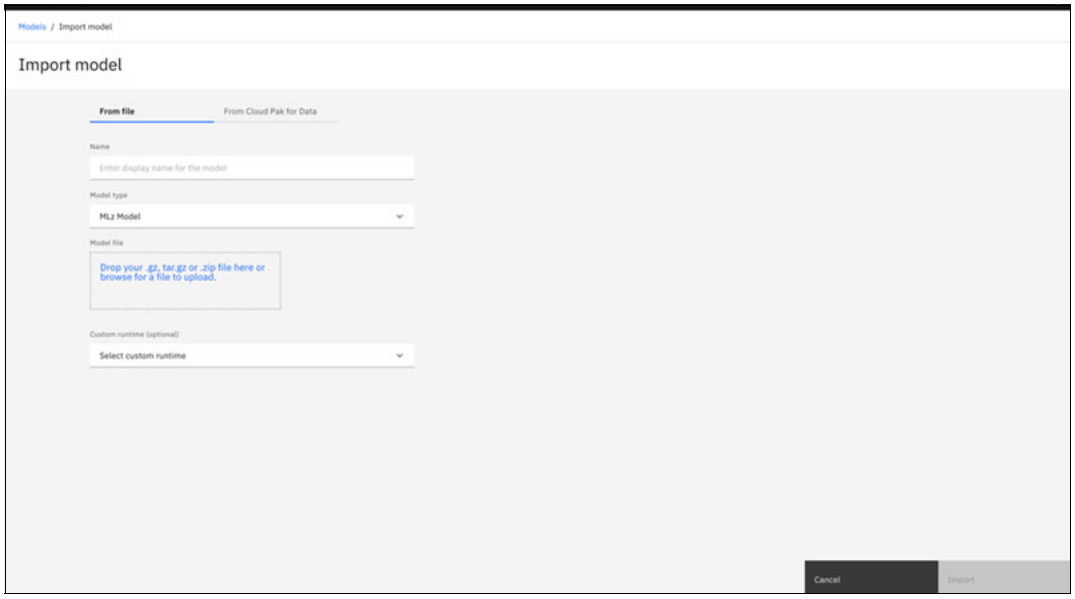


Figure 4-10 Import MLz model to MLz

Using the model utility provided by MLz, you can save Python models, such as scikit-learn and XGBoost, that were trained on any platform and save it into a format that can be imported into MLz for scoring on IBM z/OS. Thus, no matter where you decide to train your models, you have the ability to deploy them on IBM Z by converting them to ONNX or PMML formats.

For more information and a detailed procedure on importing a model from file into MLz, see:

https://www.ibm.com/docs/en/wml-for-zos/enterprise/3.1.0?topic=wmlz-importing-models-from-file

Example 4-1 shows how to convert a trained TensorFlow model to ONNX. This conversion requires running python and makes use the **tf2onnx** to perform the conversion of the deep learning model.

Example 4-1

```
# Run python command with tf2onnx to convert deep learning model to ONNX, and save
it to the local file system
!python -m tf2onnx.convert --saved-model $work_dir"fraud_cnn_model" --opset 10
--output $work_dir"model.onnx"
```

Once you have converted the model to ONNX, you can import it using the MLz UI, as shown in Figure 4-11 on page 67.

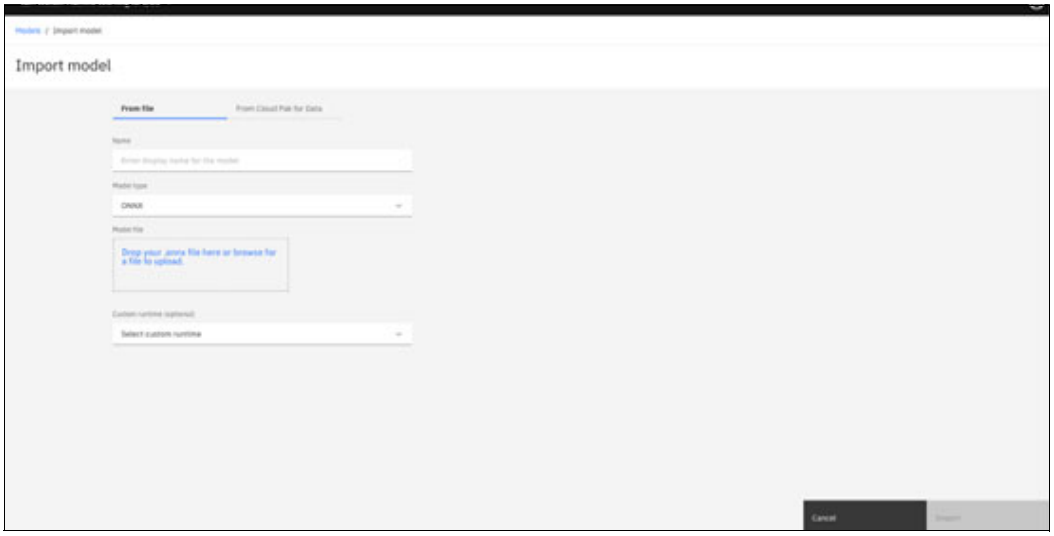


Figure 4-11 Import ONNX model to MLz

Converting a trained scikit-learn model to PMML requires a python environment to run the **`sklearn2pmml`** library. The **`sklearn2pmml`** performs the conversion with the given machine learning pipeline. The following line of code can be used to convert a trained scikit-learn model to PMML:

```
sklearn2pmml(pipeline, 'random_forest.pmml', with_repr=True)
```

Once you have converted the model to PMML, you can import it using the MLz UI, as shown in Figure 4-12 on page 67.

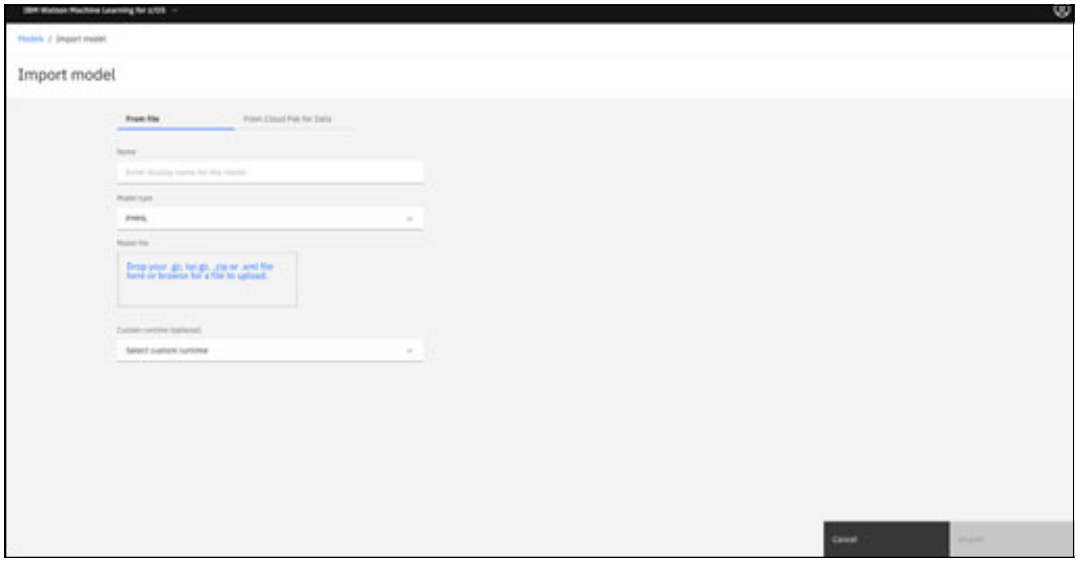


Figure 4-12 Import PMML model to MLz

Example of saving CP4D model

If you trained your model using CP4D, you can also import it into MLz using the UI, as shown in Figure 4-13 on page 68. Unlike the other model formats, CP4D is not imported from file. Instead, you need to provide various details that are specific to the CP4D instance and model, such as host name, port, username, password, deployment space, mode, and model name.

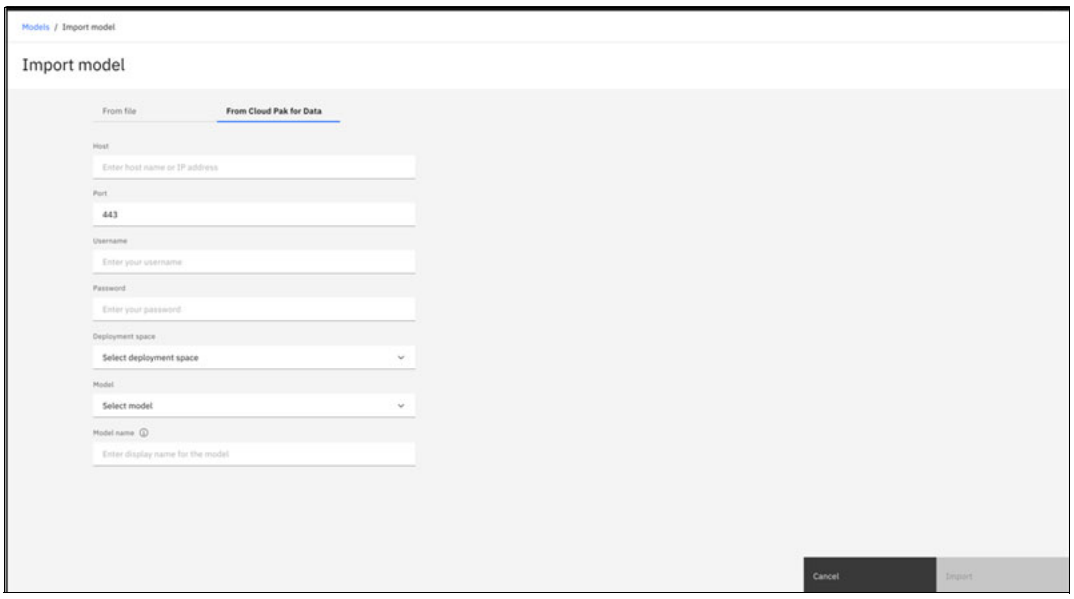


Figure 4-13 Import CP4D model to MLz

4.4.4 Evaluating model formats

As previous mentioned, MLz supports various model types.

Leveraging ONNX allows you to train your models off platform and then bring them to IBM Z by deploying with MLz. ONNX does not include preprocessing support, so depending on the use case, there may be additional work that needs to be done at inference time to account for the unprocessed data that is being presented. ONNX is a standard format so this is an advantage as it is widely supported.

Using PMML also allows you to train your models off platform, if desired. One advantage of the PMML format is that it gives you the ability to embed the preprocessing directly within the generated PMML pipeline. By doing this, it handles the preprocessing at inference time without any additional code needed to handle incoming unprocessed data.

4.4.5 Tuning model parameters

In order to improve the performance of the model, training requires hyper parameter tuning. This tuning is an experimental process. Each of the training frameworks discussed support to efficiently perform model tuning. As the number of hyper parameter variations increase, the computation and time needed to build the model increases.

4.5 Model building use cases

Once the credit card transaction data set has been cleaned, we can perform model training. You can utilize several different model frameworks and formats, but we will discuss two examples.

Use case 1: Training a TensorFlow model and saving to ONNX format

The following the steps are provided as an example workflow to building a fraud detection model using TensorFlow.

1. Build a sequential neural network model with a couple of layers as outlined in Example 4-2.

Example 4-2 Sequential neural network model with a couple of layers

```
# Build a Sequential model
model = Sequential()

# Add a 1D convolution layer
model.add(Conv1D(32, 2, activation='relu', input_shape = X_train[0].shape))
model.add(BatchNormalization())
model.add(MaxPool1D(2))
model.add(Dropout(0.2))

model.add(Conv1D(64, 2, activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool1D(2))
model.add(Dropout(0.5))

# Convert 2D data to vector
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))

# Define output data shape and activation as sigmoid
model.add(Dense(1, activation='sigmoid'))
```

2. Compile the TensorFlow model with Adam optimizer:

```
# Compile the model with Adam optimizer
model.compile(optimizer=Adam(learning_rate=0.0001), loss =
'binary_crossentropy', metrics=['accuracy'])
```

3. Train the TensorFlow model:

```
# Fit the model
model.fit(X_train, y_train, epochs=epochs, validation_data=(X_test, y_test)
```

4. Save the model:

```
# Save model to local file system
model.save(work_dir + "model")
```

5. Convert the model to ONNX:

```
# Run python command with tf2onnx to convert deep learning model to ONNX, and
save it to the local file system
!python -m tf2onnx.convert --saved-model $work_dir"fraud_cnn_model" --opset 10
--output $work_dir"model.onnx"
```

Use case 2: Training a Scikit-learn model and saving to PMML

In this example, we will build a fraud detection model using Scikit-learn by following these steps:

1. Split the training and testing data:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)
```

2. Define the preprocessing pipeline steps:

```
feature_transformers = [
   ('num', Pipeline(steps=[('normalizer', StandardScaler())]), num_feats),
   ('cat', Pipeline(steps=[('normalizer', OrdinalEncoder())]), cat_feats)
]
preprocessor = ColumnTransformer(transformers=feature_transformers)
```

3. Build pipeline with random forest model:

```
pipeline = PMMLPipeline([
   (“preprocessor”, preprocessor),
   (“classifier”, RandomForestClassifier())
])
```

4. Train model pipeline:

```
pipeline.fit(X=X_train, y=y_train)
```

5. Convert model pipeline to PMML:

```
Sklearn2pmml(pipeline, 'random_forest.pmml', with_repr=True)
```

5

Model deployment and inferencing

Model deployment and inferencing are crucial steps in the machine learning and artificial intelligence pipeline. Model deployment involves taking a trained machine learning or deep learning model and making it accessible for real-world applications. It is the process of deploying the model to a production environment where it can serve predictions or make decisions based on new, incoming data. Inferencing, on the other hand, is the act of using the deployed model to make predictions or perform tasks on unseen data. It is the stage where the model applies its learned knowledge to make real-time decisions, such as image recognition, natural language processing, or anomaly detection, providing valuable insights and automation in various industries and applications.

Proper deployment and efficient inferencing are critical for the successful integration of AI and machine learning into practical solutions. In the next sections, we will go over these topics in details.

5.1 Model deployment environments and advantages of mainframe systems

In today's data-driven world, machine learning models and algorithms are at the heart of decision-making processes. Deploying these models effectively is essential for businesses and organizations to leverage their predictive power. Model deployment can take place in various environments. Mainframe systems, often associated with legacy technology, continue to play a pivotal role in modern computing environments due to several key advantages. Recognizing the enduring advantages of Mainframe systems is essential for making informed decisions in the ever-evolving landscape of technology and data management.

5.1.1 Model deployment

In the world of artificial intelligence and machine learning, deploying models effectively is a crucial step towards turning insights into tangible results. The choice of model deployment environment plays a pivotal role in determining the success of these endeavors. While various options exist, one environment that offers unique advantages is the venerable mainframe system.

In the realm of artificial intelligence (AI), building powerful machine learning models is just the beginning. The goal is to deploy these models effectively, ensuring they perform optimally in real-world scenarios. Model deployment environments play a crucial role in this process, providing the necessary infrastructure and resources to unleash the potential of AI.

Importance of model deployment environments

The right deployment environment provides the infrastructure, scalability, flexibility, real-time responsiveness, data security, and collaboration capabilities necessary for successful model deployment.

By carefully considering and selecting the appropriate deployment environment, organizations can unlock the full potential of their AI initiatives, delivering accurate predictions, efficient decision-making, and transformative outcomes. As AI continues to reshape industries, understanding and prioritizing the importance of model deployment environments will be a key factor in driving AI-driven success.

These below list the importance of model deployment environments and highlight the key reasons why organizations should give careful consideration to this critical aspect of AI implementation.

Infrastructure and resources

Model deployment environments provide the necessary infrastructure and resources to support the execution of machine learning models. These environments encompass a range of options, including on-premises infrastructure or cloud-based solutions. The chosen environment should align with the specific needs of the organization, considering factors such as scalability, performance, data storage, and computational power. The right infrastructure ensures models can handle large data sets, process complex computations efficiently, and deliver accurate results in a timely manner.

Scalability and flexibility

Successful AI implementation requires the ability to scale models as the business grows and demands change. Model deployment environments that offer scalability and flexibility enable organizations to adapt to evolving needs. Whether it is an increase in data volume, a growing user base, or expanding use cases, the deployment environment should accommodate these

changes seamlessly. Scalable environments allow for the deployment of larger models, handling greater computational demands, and catering to increased workloads without compromising performance.

Real-time responsiveness

Certain applications require real-time responsiveness, where decisions need to be made and actions taken instantaneously. Model deployment environments that support low-latency and high-throughput inference enable real-time decision-making. For time-critical scenarios, such as fraud detection, autonomous vehicles, or financial transactions, the deployment environment must provide the necessary computational power to process data swiftly and deliver timely predictions or actions.

Data security and compliance

Data security and compliance are paramount in AI implementations, especially when handling sensitive information. Model deployment environments should provide robust security measures to protect data throughout the deployment process. This includes secure data storage, encryption mechanisms, access controls, and compliance with relevant regulations. Choosing a deployment environment that prioritizes data security ensures that confidential information remains protected, mitigating the risk of data breaches and maintaining compliance with privacy regulations.

AI model deployment options: on-premises and cloud

To unleash the power of AI, organizations must carefully consider the deployment options available. Two prominent options are on-premises and cloud-based deployments. Each approach comes with its own set of advantages and considerations, and the choice between the two depends on the specific needs and goals of the organization.

The decision between on-premises and cloud-based deployments depends on factors such as data privacy requirements, existing infrastructure, budget constraints, and the organization's overall AI strategy. Some organizations may opt for a hybrid approach, leveraging both on-premises and cloud resources to strike a balance between control, security, and scalability.

Successful AI deployment requires a thoughtful evaluation of the available deployment options, understanding the specific needs of the organization, and aligning the chosen approach with the organization's long-term AI goals and strategies.

Consideration for on-premises deployment

On-premises deployment refers to hosting AI models within an organization's own infrastructure, typically within their data centers. For the advantages and considerations of this deployment option:

- Infrastructure costs - On-premises deployment requires up front investments in infrastructure, including hardware, networking, and maintenance. Organizations must carefully consider these costs before opting for this deployment option.
- Scalability challenges - Scaling on-premises infrastructure can be complex and time-consuming. Organizations need to anticipate future growth and ensure their infrastructure can handle increasing workloads and data volumes.
- Enhanced data control - On-premises deployment offers organizations complete control over their data. This can be crucial for industries with stringent data privacy and compliance requirements, as it allows organizations to maintain sensitive data within their secure infrastructure.
- Lower latency and increased performance - By hosting AI models on-premises, organizations can reduce latency and achieve faster response times. This is particularly

important for real-time applications that require immediate decision-making or low-latency predictions.

Consideration for cloud-based deployment

Cloud-based deployment refers to providing scalability, cost efficiency, and accessibility. Organizations should carefully weigh the advantages and considerations of each option, considering their specific requirements, data sensitivity, performance needs, and cost constraints.

- Data security and compliance - Organizations must carefully evaluate the security measures provided by cloud service providers and ensure they meet their specific data security and compliance requirements. This includes considerations such as data encryption, access controls, and regulatory compliance.
- Latency and network dependency - Cloud-based deployments rely on network connectivity, which can introduce latency. For certain real-time applications with stringent latency requirements, on-premises deployment may be more suitable.

  Organizations should carefully weigh the advantages and considerations of each option, considering their specific requirements, data sensitivity, performance needs, and cost constraints. By selecting the optimal deployment option, organizations can effectively deploy AI models and unlock the transformative potential of AI in their respective industries.

5.1.2 Drivers for mainframe deployment

Mainframe deployment provides significant advantages, including scalability, performance, reliability, security, and integration capabilities. Organizations can leverage these benefits to achieve optimal performance, ensure high availability, maintain data security and compliance, seamlessly integrate with existing infrastructure, and achieve cost efficiencies.

Mainframes continue to be a strategic choice for organizations looking to harness the power of these robust systems and drive successful and resilient applications in transaction-oriented and data-intensive environments.

Mainframes have a deep integration with existing enterprise infrastructure, including legacy systems, databases, and applications. Deploying applications on mainframes ensures compatibility and seamless integration with the existing ecosystem. This integration simplifies data access, allows organizations to leverage their existing technology investments, and reduces complexities associated with deploying applications in disparate environments. By utilizing the integration capabilities of mainframes, organizations can modernize their legacy systems and enhance operational efficiency. Then we can explore the advantages that arise from deploying machine learning models on mainframe systems.

Large-scale and low-latency in-transaction inference

One of the key drivers for deploying machine learning models on mainframe systems is the ability to achieve large-scale, low-latency in-transaction inference. Mainframe systems equipped with specialized AI accelerators, such as those found in the Telum processor, provide a significant advantage in handling high-volume transaction processing.

These AI accelerators are purpose-built to execute AI computations efficiently, enabling organizations to perform large-scale inferencing within the transactional context. This capability is particularly valuable in scenarios where real-time decision-making and immediate responses are critical, such as financial transactions, fraud detection, or dynamic pricing.

By harnessing the computing power and AI accelerators of mainframes, organizations can ensure that their models deliver fast, accurate, and real-time insights, allowing them to respond to rapidly changing conditions and make informed decisions in the moment.

Real-time decision-making and immediate response advantages

Deploying machine learning models on mainframes provides organizations with a competitive edge in real-time decision-making. Mainframes offer the computational power and low-latency capabilities necessary for real-time data processing and analysis. This advantage is especially crucial in time-sensitive applications, where immediate responses can make a significant impact on business outcomes.

For example, in the banking sector, real-time fraud detection is critical to prevent financial losses. By deploying fraud detection models on mainframes, organizations can analyze transaction data in real-time, identify suspicious patterns, and take immediate action to mitigate potential fraud.

Similarly, in the health care domain, real-time analysis of patient data can enable prompt diagnosis and treatment decisions. The immediate response capabilities of mainframes empower organizations to make accurate, data-driven decisions and respond swiftly to emerging situations, enhancing operational efficiency and customer satisfaction.

Advantages of AI accelerator on Telum processor for efficient inference

The z/OS AI Accelerator on the Telum processor is a powerful hardware accelerator specifically designed for AI workloads on IBM z/OS mainframe systems. The Telum processor incorporates advanced technologies and optimizations to deliver exceptional performance for AI inference and training tasks.

The z/OS AI Accelerator on the Telum processor provides specialized hardware acceleration for deep learning workloads on the IBM z/OS mainframe platform. It incorporates optimizations specifically designed to enhance the performance and efficiency of deep learning tasks.

By leveraging the capabilities of the z/OS AI Accelerator, organizations can train and deploy deep learning models on their mainframe systems. This includes tasks such as training large-scale neural networks, optimizing model performance, and conducting inference for real-time predictions. The accelerator's high performance and scalability enable organizations to handle complex deep learning workloads efficiently within the mainframe environment.

Use cases for ML on IBM z/OS

Indeed, there are numerous use cases for machine learning on z/OS, showcasing the platform's versatility and its potential to revolutionize various industries. Let us explore some of the most compelling and impactful use cases.

With its robust and secure infrastructure, z/OS empowers organizations to unlock the potential of AI and gain a competitive edge in today's data-driven world. As AI technology continues to advance, we can expect even more innovative and transformative applications of machine learning on z/OS across various industries, including the following:

- Real-time fraud detection

  Machine learning on z/OS can play a vital role in detecting and preventing fraudulent activities in various domains, including financial services, insurance, and retail. By analyzing transactional data, customer behaviors, and historical patterns, machine learning models can identify anomalies and detect potential fraud. The z/OS AI Accelerator enhances the performance and real-time processing capabilities of these models, enabling organizations to mitigate risks and protect against fraudulent activities more effectively.

- Anomaly detection

  Financial transactions often involve complex patterns and vast volumes of data. Deploying machine learning models on z/OS enables organizations to analyze historical transaction data and identify anomalies that could indicate potential risks. By utilizing the z/OS AI Accelerator, financial institutions can efficiently process large data sets and identify unusual patterns or behaviors, enabling them to proactively mitigate risks and safeguard against fraudulent activities or compliance breaches.

- Predictive maintenance

  In industries like manufacturing and transportation, machine learning models on z/OS can monitor equipment health and performance, predicting maintenance needs and preventing costly breakdowns. This proactive approach can optimize maintenance schedules and extend the lifespan of critical assets.

Leveraging data gravity

The data gravity of z/OS is a significant advantage for organizations deploying applications, including machine learning models, within the mainframe environment. It reduces data movement, enhances data access and performance, simplifies integration with existing infrastructure, strengthens data security and compliance, and promotes effective data governance. Leveraging z/OS data gravity allows organizations to harness the full potential of their transactional data and derive valuable insights within a trusted and centralized ecosystem.

Exploiting existing mainframe data hubs for AI model deployment

Exploiting existing mainframe data hubs for AI model deployment is a strategic approach that harnesses the power of machine learning within the mainframe environment. The mainframe has long been a cornerstone of enterprise computing, serving as a robust and reliable platform for critical business applications. With the advent of AI and machine learning, organizations are seeking ways to leverage their mainframe data and infrastructure to derive valuable insights and make data-driven decisions.

The following list describes the key features for exploiting existing mainframe data hubs:

- Data integration and data governance and compliance

  Mainframe data hubs are typically integrated with various applications, databases, and legacy systems. By utilizing these existing integrations, organizations can seamlessly access and integrate data into AI workflows without the need for extensive data transformations or migrations.

  Mainframe systems are also known for their robust data governance and compliance capabilities. By deploying AI models within the mainframe environment, organizations can leverage existing data governance frameworks, access controls, and auditing mechanisms. This ensures that AI models adhere to regulatory requirements and privacy standards. Leveraging trusted data governance practices within mainframe data hubs provides organizations with the confidence that their AI deployments are compliant, secure, and aligned with existing governance policies.

- Data proximity and reduced latency and cost consideration

  Mainframe data hubs contain vast amounts of critical transactional data that are essential for training and deploying AI models. By leveraging existing mainframe data hubs, organizations can minimize data movement and take advantage of data proximity. This reduces latency in accessing and processing data, enabling faster model training and inference. Exploiting existing mainframe data hubs for AI model deployment can offer cost efficiencies.

Since the data already resides within the mainframe environment, organizations can avoid the costs associated with data transfers and storage in separate environments. Leveraging existing mainframe infrastructure and investments reduces the need for additional hardware and infrastructure expenses, resulting in cost savings. Eliminating the need for extensive data transfers.

Eliminating the need for extensive data transfers

By deploying AI models within the mainframe environment, organizations can eliminate the need for extensive data transfers between different systems or platforms. Data transfers can be a resource-intensive and time-consuming process, especially when dealing with large volumes of data. Moving data between systems not only incurs additional costs but also introduces potential security risks and data integrity issues.

By exploiting existing mainframe data hubs for AI model deployment, organizations can keep their critical data securely within the mainframe and leverage it directly for AI model training and inference. This approach significantly reduces the overhead associated with data movement, as the AI models can directly access the relevant data stored within the mainframe.

The following list describes the key features for eliminating the extensive data transfers:

- Eliminate the requirement for extensive data transfer

  By leveraging existing mainframe data hubs for AI model deployment, organizations can eliminate the requirement for extensive data transfers between different environments. Since the data is already stored within the mainframe environment, there is no need to move large volumes of data to other platforms or cloud environments for model training or inference. This eliminates the complexities and potential delays associated with data transfers, ensuring that the AI models can access the required data without latency or integrity issues.

- Utilize the data where it resides

  Instead of duplicating or transferring data to separate environments, organizations can utilize the data where it resides, directly within the mainframe data hubs. This data proximity allows AI models to efficiently access and process the transactional data within the mainframe environment. By avoiding the need for extensive data transfers, organizations save time, reduce network bandwidth requirements, and minimize the risk of data duplication or inconsistency.

- Maintain data security and compliance

  Moreover, eliminating extensive data transfers helps maintain data security and compliance. By keeping the data within the trusted mainframe environment, organizations can leverage the existing security measures, access controls, and data governance frameworks already in place. This ensures that sensitive data remains protected and that AI model deployments adhere to regulatory requirements and privacy standards.

Overall, by eliminating the need for extensive data transfers, organizations can streamline the AI model deployment process, reduce complexities, enhance data integrity, and leverage the efficiency and security of the mainframe data hubs. This approach allows for faster and more reliable AI deployments, enabling organizations to derive insights and make informed decisions without the delays and challenges associated with moving data across different environments.

Ensuring data integrity, security, and compliance

Ensuring data integrity, security, and compliance is of paramount importance in any AI deployment, especially when working with sensitive and critical data within the mainframe

environment. The mainframe is known for its robust security features, and leveraging these capabilities is essential to maintain the confidentiality, integrity, and availability of data.

By deploying AI models within the mainframe, organizations can take advantage of the mainframe's built-in security mechanisms, such as access controls, encryption, and audit trails. These security measures help protect data from unauthorized access, modification, or disclosure, ensuring that only authorized users have the necessary permissions to interact with the data and AI models.

Data integrity

Mainframe systems have a long-standing reputation for maintaining data integrity. The mainframe data hubs store critical transactional data, which is essential for AI model training and inference. The data is protected through various mechanisms such as data redundancy, checksums, and error detection and correction codes. By deploying AI models within the mainframe environment, organizations can leverage these built-in data integrity measures, ensuring that the data used for training and inference remains accurate and consistent.

Data security

Mainframes are known for their robust security features. Organizations can rely on the existing security infrastructure of the mainframe data hubs to safeguard sensitive data used in AI model deployments. This includes features such as access controls, encryption, and secure authentication mechanisms. By utilizing these security measures, organizations can ensure that only authorized personnel have access to the AI models and the underlying data, reducing the risk of data breaches and unauthorized usage.

Compliance

The mainframe environment is designed to comply with various regulatory standards and industry-specific compliance requirements. Leveraging existing mainframe data hubs for AI model deployment ensures that organizations can adhere to regulatory guidelines such as data privacy regulations, financial compliance standards, and industry-specific compliance frameworks. The mainframe's built-in auditing capabilities and governance frameworks further support compliance efforts, providing organizations with the necessary tools to monitor and track AI model usage and ensure adherence to compliance regulations.

5.2 Deploying with ease through MLz: features and benefits

MLz is an enterprise machine learning solution specifically designed to run on IBM Z. It offers the flexibility of being deployed as a standalone solution or seamlessly integrated into your enterprise AI capabilities as a scalable platform.

ML for z/OS comes equipped with a web user interface (UI), a variety of APIs, and a comprehensive web administration dashboard. These features provide a robust suite of user-friendly tools for streamlined model development, deployment, user management, and system administration. This simplifies the machine learning operations for various key roles, including system administrators, machine learning administrators, data scientists, and application developers.

For instance, system administrators can effortlessly manage and monitor the services through the administration dashboard, ensuring their smooth operation. Machine learning administrators can deploy machine learning models as services to applications like CICS, Java, and others on the Z platform. Consequently, application developers can leverage these services in their own applications to make real-time predictions.

By utilizing Machine Learning for IBM z/OS, organizations can enhance their machine learning capabilities and harness the power of IBM Z for efficient and effective AI-driven solutions.

5.2.1 Features of MLz

MLz offers an intuitive and user-friendly interface, making it easy for both beginners and experts to navigate. MLz has advanced analytics capabilities, providing valuable insights to support data-driven decision-making. It is your go-to solution for streamlined development and seamless deployment, making it a preferred choice among developers. With MLz, you have a comprehensive framework at your fingertips, ready to accelerate your development and deployment initiatives.

User-friendly interface

A key advantage of a user-friendly MLz interface is its ability to minimize the machine learning curve for new users. Clear instructions, visual cues, and contextual help make it easier for users to familiarize themselves with the application's functionality. Intuitive interfaces reduce the need for extensive training and enable users to start utilizing the application efficiently from the outset. This not only saves time and resources but also increases user adoption rates.

The MLz user interface on z/OS empowers users to efficiently develop, deploy, and manage machine learning models. It simplifies complex processes, offers monitoring and administration functionalities, and provides an intuitive platform for users to leverage the power of IBM Machine Learning on the IBM Z platform.

Model development

MLz UI goes beyond monitoring and managing batch processing applications. It also provides a comprehensive set of tools and functionalities for model development, enabling users to harness the power of machine learning. Within the MLz UI, users have access to a range of features that facilitate the entire model development lifecycle.

One of the key capabilities offered by the MLz UI is the ability to manage machine learning models. Users can select from a variety of algorithms and techniques that best suit their specific use cases then import the models into MLz. The MLz UI also supports the integration of custom transformers. These custom transformers enable users to incorporate their own data preprocessing or feature engineering logic into the model pipeline. This flexibility empowers users to tailor the model to specific requirements and domain expertise, enhancing the model's performance and adaptability.

Moreover, the MLz UI provides essential model evaluation tools, allowing users to assess the performance of their trained models. Users can analyze key metrics such as accuracy, precision, recall, and F1-score to gauge the model's effectiveness. This evaluation process is crucial for selecting the most appropriate model for a specific use case and ensuring that it meets the desired performance standards.

As an integrated platform, MLz UI bridges the gap between model development and deployment. Once the models are trained and optimized, users can easily deploy them as services within z/OS applications using the MLz scoring engine. This seamless integration enables organizations to leverage the power of machine learning in real-time applications, driving innovation and efficiency across the enterprise.

Model deployment

The MLz UI not only empowers users to build and train machine learning models but also offers seamless integration capabilities by enabling the deployment of these models as services. With a user-friendly interface, the deployment process becomes straightforward and efficient, allowing users to leverage the power of machine learning in their existing z/OS applications.

When deploying models, users can choose from a variety of deployment targets, catering to their specific needs and application requirements. Whether it is CICS, Java, or other z/OS applications, MLz accommodates diverse deployment environments, making it a versatile solution for organizations with varying infrastructures.

The deployment process involves packaging the trained machine learning models into a format that seamlessly integrates with the target application. MLz streamlines this process, saving time and effort while ensuring compatibility and reliability. The model deployment is designed to be plug-and-play, allowing users to quickly activate the model as a service within their applications.

Once deployed, the machine learning model becomes readily accessible, providing real-time predictive capabilities to the target applications. This means that z/OS applications can harness the insights and intelligence derived from the machine learning models without requiring complex data transfers or manual interactions. The integration is seamless, enhancing the efficiency and effectiveness of existing business processes.

By enabling real-time predictions within z/OS applications, organizations can make data-driven decisions instantaneously. For example, a financial institution can deploy a credit risk assessment model within their core banking system, facilitating real-time evaluation of loan applications. Similarly, an e-commerce platform can integrate a recommendation system to deliver personalized product suggestions to customers as they browse through their online store.

The ability to deploy machine learning models as services within z/OS applications opens up new possibilities for automation, optimization, and enhanced customer experiences. It allows organizations to maximize the value of their machine learning investments, driving innovation and gaining a competitive edge in the rapidly evolving technological landscape.

Scalability and performance

MLz harnesses the power of IBM Z mainframe architecture to deliver unmatched scalability and performance for machine learning workloads. The mainframe's inherent design principles of reliability, availability, and security are seamlessly integrated into the MLz platform. The scalability of IBM Z ensures that organizations can handle increasing workloads and large-scale deployments with ease.

MLz seamlessly integrates with a wide range of ecosystem tools, libraries, and frameworks, allowing organizations to leverage their existing investments. It supports popular machine learning frameworks such as Scikit Learn, XGBoost, PMML, ONNX and Apache Spark, enabling data scientists to leverage familiar tools for model development and training. This integration simplifies the transition to MLz and accelerates time-to-value for machine learning projects.

MLz leverages the latest z16 Telum Accelerator capabilities offered by IBM Z to accelerate deep learning ONNX model inference. Enhanced performance of ONNX model inferencing by leveraging the advanced model conversion utility of IBM deep learning compiler (DLC) and the on-chip AI accelerator of IBM Telum processor on z16.

MLz simplifies the deployment of machine learning models at scale. It provides a scalable platform for deploying models as services to support real-time predictions and inference. The mainframe's robust processing capabilities allow organizations to handle high volumes of requests, ensuring smooth and uninterrupted model deployments. MLz seamlessly integrates with existing enterprise systems, enabling organizations to infuse machine learning capabilities into their critical business processes.

5.2.2 Benefits of using MLz

MLz empowers organizations to harness the full potential of machine learning in a scalable, secure, and efficient manner. By leveraging the power of IBM Z mainframe architecture, MLz provides enhanced scalability, seamless integration, comprehensive model versioning and management, advanced security, streamlined deployment and monitoring, and cost-effective resource optimization. Embracing MLz enables organizations to accelerate innovation, make data-driven decisions, and drive business success in the era of machine learning and artificial intelligence.

MLz also offers a comprehensive set of features and capabilities designed to empower organizations in their machine learning endeavors. By leveraging the power of IBM Z mainframe architecture, MLz provides a range of benefits that enable organizations to accelerate model development, improve operational efficiency, and drive business success. In this section, we will explore the key benefits of using MLz.

Simplified deployment process

MLz simplifies the process of deploying machine learning models into production environments. It provides a scalable platform for deploying models as services, allowing real-time predictions and inference. With MLz, organizations can seamlessly integrate machine learning capabilities into their existing applications and systems. The platform also offers monitoring and performance tracking features, enabling organizations to monitor model performance, detect anomalies, and make necessary adjustments for continuous improvement.

MLz offers a web-based user interface that simplifies the management and monitoring of deployed models. The intuitive interface allows users to easily deploy, monitor, and control their models. From managing model versions to tracking performance metrics, the user-friendly interface provides a centralized platform for efficient model deployment management.

MLz supports continuous integration and delivery practices, enabling organizations to automate the deployment process. By integrating MLz with CI/CD pipelines, organizations can automate model packaging, testing, and deployment, ensuring a streamlined and efficient deployment process. MLz provides comprehensive REST APIs for model and deployment management which can be integrated into modern CI/CD pipeline.CI/CD integration with MLz reduces manual efforts, improves consistency, and accelerates time-to-market for machine learning applications.

We can ascertain the MLz model management lifecycle is a comprehensive and well-structured process that enables organizations to effectively develop, deploy, monitor, and update machine learning models on z/OS. This lifecycle ensures that models are continuously optimized and leveraged to their fullest potential, driving business value and staying ahead of the competition.

Overall, the MLz model management lifecycle ensures that machine learning models are developed, deployed, and monitored efficiently, empowering organizations to harness the power of AI on the z/OS platform effectively. By following this structured approach,

businesses can drive innovation, optimize processes, and make data-driven decisions, leading to improved operational efficiency and competitive advantage.

Secure and trusted environment

To establish a secure and trusted environment for MLz (assuming it is a specific application or system), you can consider the following guidelines:

- Authentication and authorization

  MLz implements a robust authentication mechanism to ensure that only authorized users can access the MLz application. For users, it must be required to create an IBM RACF® keyring for user authentication in WML for z/OS and then configure a keyring keystore (JCERACFKS) to use the keyring for authentication. The protection does not only limit to the tradition password but also supporting the user certification-based request to MLz authentication API. Create RACF access control rules to govern access to MLz resources. Specify who can access which resources and what actions they can perform. This includes defining rules for read, write, execute, or administrative access based on resource classes and user profiles.

- Secure communication

  Protect the communication channels used by MLz with Transport Layer Security (TLS) or Secure Sockets Layer (SSL) encryption protocols. This ensures that data transmitted between users and the application remains confidential and secure. Besides this, you can further strengthen the security of your network connections by leveraging the Application Transparent Transport Layer Security (AT-TLS) capability on z/OS. With the AT-TLS support, your MLz services can communicate securely with your client applications by utilizing the z/OS policy-based TLS/SSL protection. You can also enable TLS client authentication for z/OS Spark by leveraging AT-TLS.

Integration with mainframe ecosystem z16 Accelerator

The z16 Accelerator is a powerful hardware feature available on IBM z/OS systems. It offers specialized processing capabilities, such as enhanced vector instructions and advanced cryptographic functions. By leveraging the z16 Accelerator, MLz can benefit from improved performance and accelerated computations for ONNX model of deep learning.

To use the AI accelerator, you can select a scoring server that runs on z16 and a version of a model that is imported with zDNN available on your system. The z16 Accelerator provides hardware acceleration and optimized processing capabilities, which can significantly enhance the performance of machine learning workloads, including those based on ONNX models.

The integration of MLz with the z16 Accelerator enables the efficient execution of ONNX models, allowing organizations to leverage the power of their mainframe infrastructure for high-performance machine learning applications. Exploits the new Telum chip in the z16 to further enhance performance of scoring services. Realize true AI at scale by scoring every transaction with low single digit millisecond response times.

5.3 Model inferencing

Machine learning inferencing, also referred to as scoring, is the phase in the machine learning life cycle where a trained model is utilized to make predictions or classifications on new, unseen data. This stage represents the practical application of the knowledge and patterns that the model has learned during its training process. When new data is presented to the model, it employs the insights it has gained from the training data to provide meaningful and informed outputs. This process involves feeding the input data through the model's

algorithms, which then produce predictions, classifications, or scores based on the relationships it has identified. Machine learning inferencing plays a pivotal role in enabling automated decision-making, real-time recommendations, and the extraction of valuable insights from raw data. It is the culmination of the model's training efforts, transforming it from a learning algorithm into a practical tool that can contribute to a wide array of applications, from fraud detection and image recognition to natural language processing and autonomous driving.

The idea of real-time, in-transaction analytics brings a big change in how companies use their old records. It starts a new way of making decisions using data. With this approach, businesses can find new ideas, make prediction about the future, and choose what to do quickly. It is different from just looking back at old reports. Instead of focusing on the past, they look at data as it happens. This helps them predict what might happen later on. This helps a lot in many industries. It means they can give customers advise, prevent problems and frauds, sell more things, gather customers insights, manage products better, and even change how they compete with other companies. MLz can help businesses achieve those goals. It can help clients to use real-time information to make smart predictions and gain business right away.

5.3.1 Making online predictions

The significance of machine learning scoring features becomes most evident when AI models are applied to predict future outcomes, particularly in the realm of real-time business insights. Industries such as payment fraud detection, loan approval, and client onboarding greatly benefit from this application. By implementing machine learning scoring, enterprises have the capacity to evaluate the significance of transactions and allocate scores directly within the platform where these transactions take place. This approach has significant benefits for IBM Z customers. It not only ensures a heightened level of security but also results in improved overall performance. Moreover, the incorporation of machine learning scoring allows companies to make the most of their IBM Z existing infrastructure, optimizing resources and enhancing operational efficiency. In essence, the utilization of machine learning scoring features within AI models is pivotal for driving accurate predictions and informed decision-making in various critical domains, ultimately contributing to a more secure, efficient, and insightful business environment.

MLz outlines a typical machine learning life cycle within the context of machine learning. The process includes various stages, beginning with data preparation and ingestion. Data, whether sourced from historical databases like DB2 Z or imported from open-source realms, is readied for analysis. Subsequently, the model undergoes training to enhance its capabilities. Upon completion, the model enters the deployment and scoring phase, where users can choose from a range of interfaces provided by MLz. These interfaces cater to diverse application types, including options like WebSphere Application Server, CICS, IMS, or IBM Z batch job, which facilitate the processing of business logic. The chart also emphasizes several key advantages offered by MLz, such as accelerating AI model development by enabling training anywhere while ensuring scoring on z/OS. Additionally, it offers AI model lifecycle management, allowing for seamless integration and maintenance on IBM Z. The ability MLz has to infuse real-time AI model inferencing into z/OS applications can help customers in delivering dynamic insights for their enterprise applications.

The flexibility of MLz shines through its capability to facilitate training anywhere and subsequently deploy for inferencing within the MLz framework. This seamless integration accommodates models trained both on z/OS and off z/OS platforms. Through leveraging the MLz integrated Jupyter notebook, training can be conducted using data sourced directly from z/OS. This platform further supports various model development tools such as IBM CP4D. Notably, MLz accommodates an array of model types, including SparkML models,

Python-based models like Scikit-learn, XGBoost, and Arima, PMML models, and even deep learning models in ONNX format. It empowers users to harness a diverse range of model types to fulfill their inferencing needs.

The adoption of ONNX (Open Neural Network Exchange) as an open standard signifies its pivotal role in facilitating seamless framework interoperability. This standard offers a remarkable advantage, allowing for the conversion of models from frameworks like TensorFlow, Caffe, and Keras to ONNX format through the framework's provided APIs. A noteworthy feature of MLz lies in its capability to readily import deep learning models in the ONNX format, rendering them suitable for deployment to accomplish real-time inferencing tasks. To compile ONNX model into executables for deployment and inference, MLz uses IBM zDLC. It is important to note that this compilation process requires either zCX or Linux on Z.

Once the ONNX model is prepared, its scoring runs natively on the z/OS platform within the Liberty server environment. Because it is Java based, it qualifies for zIIP (IBM Z Integrated Information Processor) which is cost effective for customers. In addition, ONNX scoring gains a substantial performance boost, especially benefiting from the on-chip AI accelerator featured in the z16 architecture. Furthermore, ONNX model inferencing accommodates micro-batching scoring, catering to transactions with higher throughput. This approach ensures that the platform can effectively manage high volumes of transactions, reflecting its responsiveness to diverse business demands. Overall, the integration of ONNX within the MLz framework showcases a meticulously designed and robust architecture, facilitating seamless model deployment, inferencing, and optimization on the z/OS platform. Refer to Figure 5-1 for a detailed overview of the ONNX model inferencing.

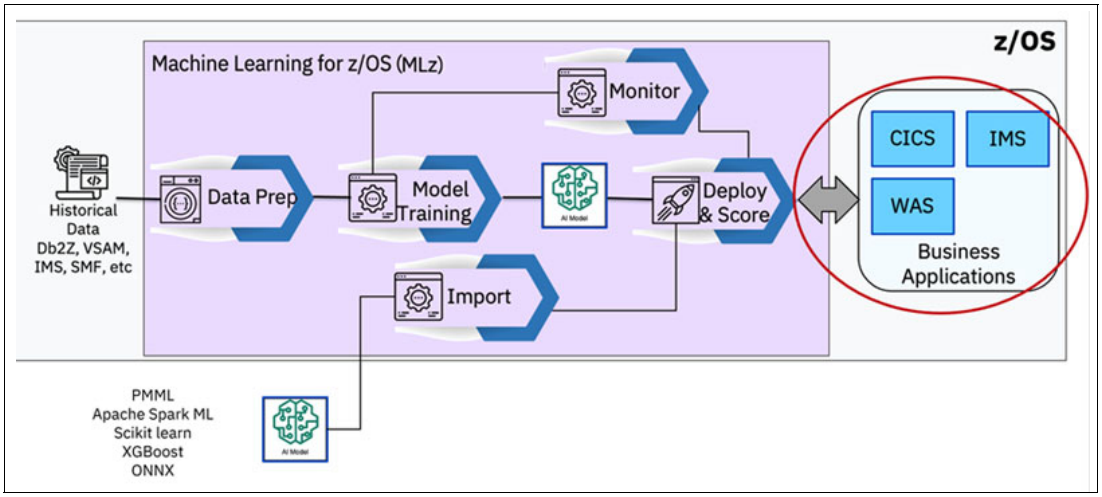


Figure 5-1 ONNX model inferencing

The MLz scoring server offers easy-to-use inferencing interfaces for z/OS developers, including REST, Java, CICS, and WOLA. It allows scoring within ongoing transactions, maintaining z/OS performance and resiliency standards. Impressively, it achieves quick scores in single-digit milliseconds, enabling rapid decision-making and seamless integration into operations. See “Best practice on model deployment and inference” on page 110 for more tips on choosing the right inferencing interfaces for your use case.

5.3.2 Interfaces of MLz online scoring service

Illustrated in Figure 5-2 on page 85 are two distinct types of interfaces offered by the MLz online scoring service. One common interface type is RESTful, extending support to a variety of model types, including SparkML, PMML, ONNX, and Python models. This interface uses

JWT token authentication for security. When an application interacts with this interface, it uses a deployment ID and URL to locate the intended model for inferencing. Input data, such as the record to be scored, are submitted in JSON. The scoring output is also presented in JSON format. The actual scoring process takes place within the MLz scoring server (Liberty Server), where Predictive models compute the result. For applications utilizing Python models, the MLz Python scoring server (Gunicorn Server) is employed instead, tailoring the approach to the specific requirements of Python-based models.

An alternate inferencing interface option is the Java interface. It supports SparkML, PMML, and ONNX models. This interface was implemented with the intention of facilitating integration with rule-based engines like IBM Operational Decision Manager (ODM). Particularly advantageous for scenarios involving transaction processing on the Liberty/WAS server, this interface enables the MLz scoring service to run on the same Liberty/WAS server. This proximity empowers applications to directly invoke the MLz scoring service through a Java native API, eliminating the need for RESTful API interactions. This approach not only fosters a closer integration between components but also enhances performance, ensuring a more efficient and streamlined scoring process.

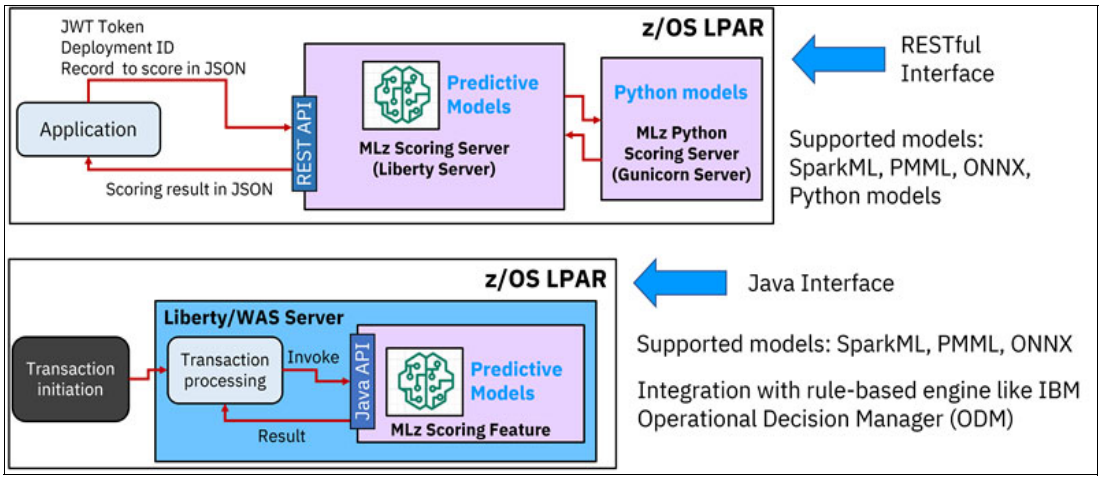


Figure 5-2 WOLA scoring interface

WebSphere Optimized Local Adapters (WOLA) is a feature within IBM WebSphere Application Server for z/OS Liberty. It plays a crucial role in managing communication between the Liberty for z/OS server and external address spaces, like CICS, Batch, or IMS, residing within the same logical partition. By configuring the scoring service with enabled WOLA functionality, you can seamlessly conduct online scoring operations for SparkML, PMML, and ONNX models. This can be achieved by utilizing the native WOLA APIs in your COBOL programs, which can run in CICS, IMS, or even in batch jobs.

Illustrated in Figure 5-3 on page 86 is the WOLA scoring interface, specifically designed for IBM Z native applications. A top advantage of this interface is its capability for memory-to-memory data exchange between COBOL applications-operating within batch Jobs, IMS, or CICS-and the MLz scoring server. Unlike traditional network-layer exchanges, this method enhances efficiency and security. Leveraging native callable interfaces provided by WOLA APIs minimizes the impact on existing COBOL applications and the work of IBM Z application programmers. This streamlined approach aligns with the system's architecture, ensuring a smooth and secure exchange of information while optimizing performance and maintaining compatibility with established processes.

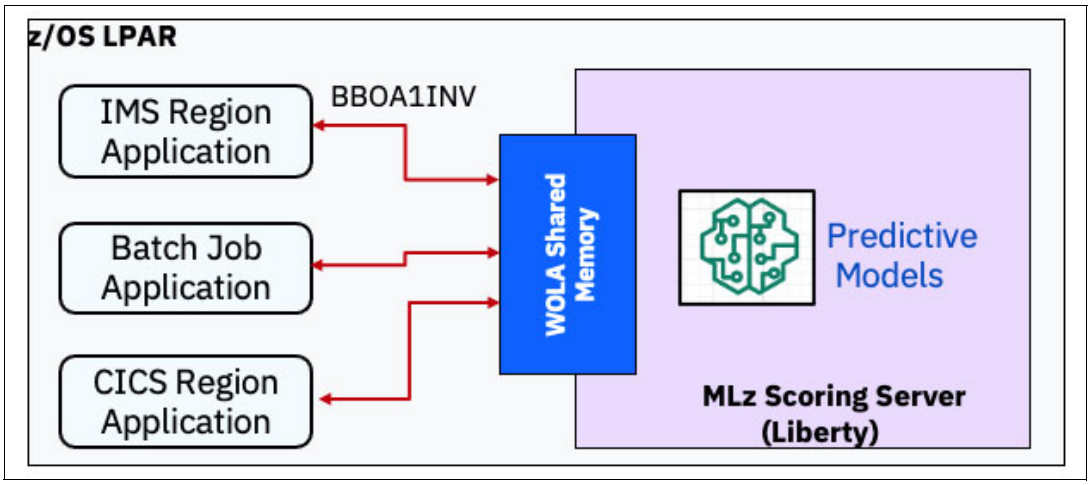


Figure 5-3 Example of using WOLA interface

Figure 5-4 on page 87 shows an example of scoring using WOLA interface with a COBOL application. It uses BBOA1INV which is a native callable inference provided by WOLA APIs. Input parameters consists of `rqst-area-addr` and `service-name`. The `rqst-area-addr` parameter serves as a pointer to the data destined for transmission to the scoring server. It includes details such as the model deployment ID, Java helper class name for model input and output, and the actual model input record. Another input parameter is the `service-name` with a value of `java:global/scoring-wola/WOLAHandler!com.ibm.ml.scoring.online.service.WOLAHandler` in our example.

```
* CHURN model
 01 CINPUT.
    03 DEPLOYMENT-ID                    PIC X(40).
    03 INPUT-CLASS                      PIC X(64).
    03 OUTPUT-CLASS                     PIC X(64).
    03  CHURNIN.
      06 EDUCATION                      PIC S9(18) COMP-5 SYNC.
      06 AGE                            PIC S9(18) COMP-5 SYNC.
      06 SEX-length                     PIC S9999 COMP-5 SYNC.
      06 SEX                            PIC X(255).
      06 NEGTWEETS                      PIC S9(18) COMP-5 SYNC.
      06 INCOME                         COMP-2 SYNC.
      06 ACTIVITY                       PIC S9(18) COMP-5 SYNC.
      06 STATE-length                   PIC S9999 COMP-5 SYNC.
      06 STATE                          PIC X(255).
 01 COUT.
        06 probability OCCURS 2              COMP-2 SYNC.
        06 prediction                        COMP-2 SYNC.
- - - - - - - - - - - - - - - - - - -   10 Line(s) not Displayed
     MOVE 'df72b722-84d3-4e83-b381-f822a8b228a9' TO DEPLOYMENT-ID.
     MOVE 'ChurnInWrapper' TO INPUT-CLASS.
     MOVE 'ChurnOutWrapper' TO OUTPUT-CLASS.
- - - - - - - - - - - - - - - - - - - -  4 Line(s) not Displayed
     MOVE 1             TO EDUCATION.
     MOVE 20            TO AGE.
     MOVE 'F'           TO SEX.
     MOVE 13            TO NEGTWEETS.
     MOVE 16552         TO INCOME.
     MOVE 1             TO ACTIVITY.
     MOVE 'ND'          TO STATE.
     MOVE 1             TO SEX-length.
     MOVE 2             TO STATE-length.
*
     MOVE 'This is a test message'       TO text-msg.
     MOVE 'java:global/scoring-wola/WOLAHandler!com.ibm.ml.scoring
-      '.online.service.WOLAHandler'
          TO service-name.

- - - - - - - - - - - - - - - - - - -   36 Line(s) not Displayed
       SET rqst-area-addr TO ADDRESS OF CINPUT
       SET resp-area-addr TO ADDRESS OF COUT

       CALL 'BBOA1INV' USING
           register-name,
           rqst-type,
           service-name,
           service-name-length,
           rqst-area-addr,
           rqst-area-length,
           resp-area-addr,
           resp-area-length,
           wait-time,
           rc,
           rsn,
           rv
```

Figure 5-4 Example of using native CICS interface

Upon calling the MLz WOLA scoring service, an output parameter named `resp-area-addr` comes into play. This parameter acts as a pointer to the data area responsible for carrying the scoring output, which shows the probability predication for the model. Through this COBOL example, the process of scoring using WOLA APIs becomes clearer, demonstrating the input and output parameters and interaction with the MLz scoring service. Native IBM Z application developers do not need to deal with JSON or HTTP requests associated with RESTful APIs.

Through the CICS interface, you can initiate online scoring for SparkML, PMML, and ONNX models (see Figure 5-5 on page 88). In the context of a CICS region, a Liberty server operates under the name ALNSCORE. When you want to perform online scoring involving SparkML, PMML, and ONNX models within your CICS COBOL application, the `CICS LINK` command comes into play. By utilizing this command, you can efficiently invoke the ALNSCORE interface, facilitating seamless online scoring operations. This setup brings a simplified programming approach for CICS COBOL applications, making it easier to integrate model inferencing. Additionally, this approach yields enhanced performance in comparison to using the RESTful interface.

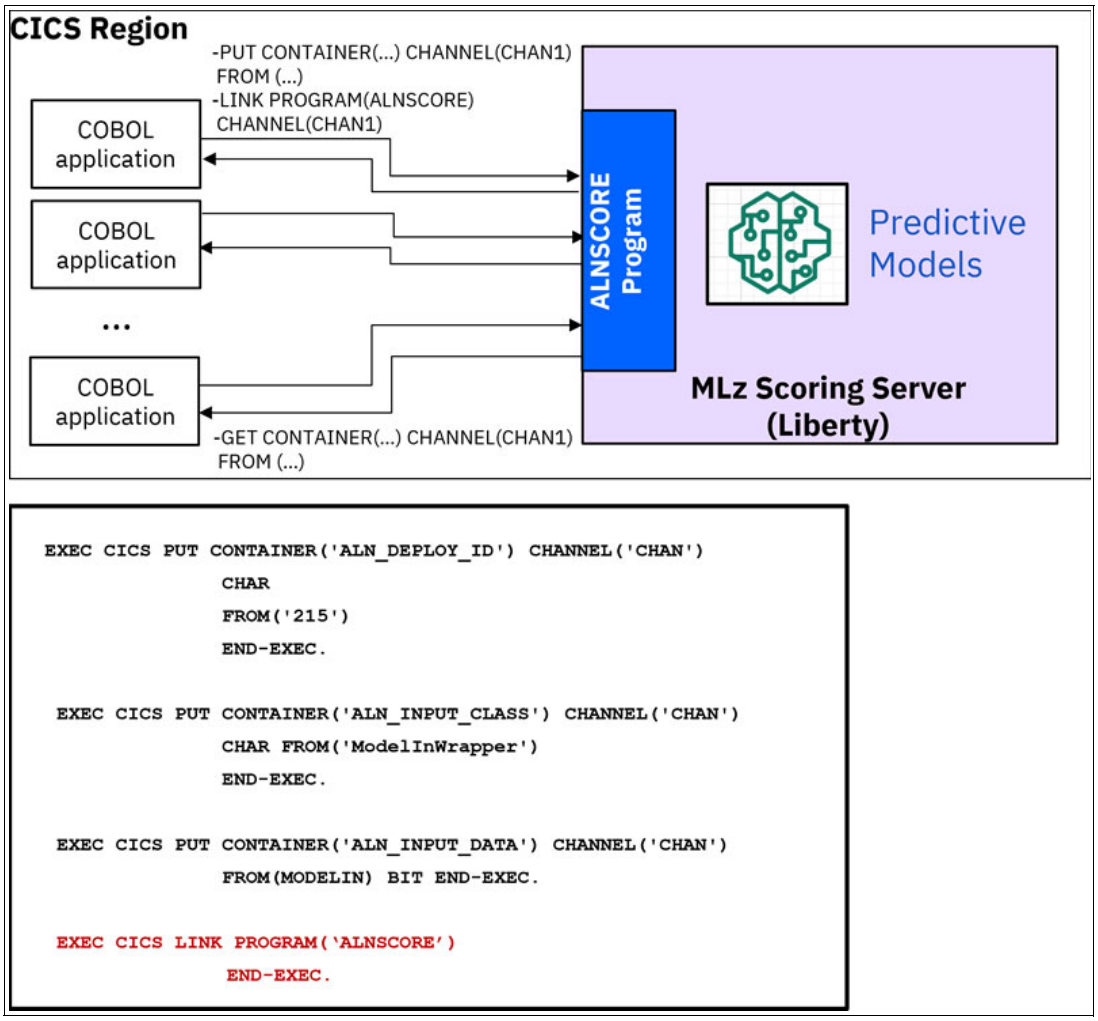


Figure 5-5 CICS region

5.4 Sample use case for MLz model deployment

MLz is designed to support a wide range of use cases and scenarios, providing flexibility and versatility for various needs. With its robust capabilities, MLz can cater to diverse user requirements, enabling organizations to address multiple use cases effectively. Whether it is developing and deploying machine learning models, performing advanced analytics, or leveraging AI technologies, MLz offers the necessary tools and features to support a broad spectrum of user scenarios. Its adaptability and comprehensive functionality make it a valuable solution for a variety of use cases, empowering users to leverage the power of machine learning and AI in their specific domains.

Today, we will delve into a specific use case involving the prediction of z/OS-specified batch monitor of job elapsed time.

First, why is the efficiency of batch jobs crucial?

A single host typically runs between 10,000 to 60,000 jobs per day, operating 24 hours a day. These different batch jobs may have interdependencies, and even a slight oversight can lead

to repetitive work and reduced efficiency. Therefore, efficient batch processing is vital to meet the high productivity demands of enterprises.

For businesses in the financial industry such as banks, tasks like salary payments, mortgage repayments, interest accruals, and other operations are usually processed in nightly batch jobs after regular business hours. Ensuring that these nightly batch jobs are completed within specified time windows is critical for the normal operation and continuity of banking transactions and services during the day.

Secondly, there is no effective scientific method for monitoring and predicting the progress of batch processing jobs currently.

Post-analysis, job re-running in case of failures, and estimation of batch job durations based on experience.

Here, we will explore how this use case can be effectively applied using MLz. By examining this scenario, we aim to understand the practical implementation of MLz in predicting the elapsed time of batch Job on the z/OS platform. Through this exploration, we will gain insights into the seamless integration of MLz into the mainframe environment, showcasing its capabilities in optimizing and enhancing performance for batch job processing on z/OS.

5.4.1 The use case for batch job elapsed time prediction

Batch job elapsed time prediction is a valuable case that allows you to estimate the time it will take for a batch job to complete its execution. This predictive capability is based on historical data, job complexity, and various other factors. By providing insights into job duration, it enables better planning, resource allocation, and decision-making within your workflow or system. With batch job elapsed time prediction, you can optimize your operations and enhance overall efficiency.

Business questions and personas

Business questions and personas are essential components of a successful use case. They provide the foundation for building meaningful and impactful solutions that address real business needs. By keeping these aspects in mind, organizations can develop data-driven strategies and make informed decisions that drive growth and success.

Business questions for batch monitor

In today's fast-paced business environment, time is of the essence. For z/OS organizations relying on batch applications for critical data processing, estimating the time required for these applications to complete their tasks is essential for efficient planning and decision-making. In this section, we will explore how advanced analytics and predictive modeling techniques can help address two crucial business questions:

- How long will it take for my z/OS batch applications to complete today?

  Batch applications play a vital role in processing large volumes of data in a scheduled manner. Accurate estimation of the time required for these applications to finish their tasks allows organizations to manage resources effectively, optimize workflows, and meet operational deadlines. By leveraging predictive modeling and the power of data, businesses can gain insights into the expected run times of their batch applications, enabling them to make informed decisions and allocate resources accordingly.

  In a mainframe system, approximately 10,000 to 60,000 jobs run continuously 24 hours a day. These jobs are interconnected, and any oversight can lead to redundant work and decreased efficiency. Therefore, efficient batch processing is crucial for maintaining high productivity in an enterprise. For businesses in industries like banking and finance, tasks

such as salary payments, mortgage repayments, interest calculations, and other operations are typically handled by overnight batch processing after the close of business hours. Ensuring that the overnight batch jobs are completed within the designated time window is of utmost importance for the smooth functioning and operation of daily transactions and business activities in banks.

IBM z/OS organizations that invest in developing accurate completion time prediction models for their batch applications can gain a competitive advantage in today's fast-paced business landscape. By optimizing workflow efficiency and resource allocation, businesses can achieve enhanced operational performance, improve customer satisfaction, and drive overall success.

- How long will it take for z/OS batch application to complete after a specific job finishes?

  In complex data processing environments, multiple batch applications are often interconnected, with certain jobs acting as prerequisites for subsequent tasks. Knowing the estimated completion times of these subsequent applications is essential for maintaining smooth operations and meeting critical business requirements. Accurate estimation enables organizations to manage dependencies, schedule jobs effectively, and optimize resource allocation, ultimately minimizing downtime and maximizing workflow efficiency.

  Estimating batch application completion times after a specific job involves leveraging historical data, considering dependencies, and utilizing predictive modeling techniques.

Job Name	Start Time	End Time	Elapsed Time (min)
Job1	1:00	1:02	2
Job2	1:00	1:03	3
Job3	1:10	1:11	1

Figure 5-6 Batch application with sample jobs running example

Based on the outlined workflow Figure 5-6, we can assume that the z/OS batch application consists of three key milestone jobs. These milestone jobs serve as significant stages within the overall batch processing flow. Additionally, considering the available data, we will estimate the total execution time for the batch application. See Example 5-1 for a batch execution of 11 minutes elapsed time.

Example 5-1 Elapsed Time Batch Execution Time

```
Elapsed Time with Batch Execution Time=
max(End Time)-min(Start Time)=1:11-1:00=11min
```

After assigning start and end times to each individual job, each job within the batch application runs for a specific duration. Through continuous monitoring of the z/OS batch application's progress, we can make predictions about the expected end time for each job, including job1, job2, and job3. If the predicted end time for job2 deviates from the expected time frame, it may indicate potential issues specific to job2. Detecting such deviations allows system administrators to receive immediate alerts, enabling them to promptly investigate and address any underlying problems, without waiting for subsequent jobs to complete. Our objective extends beyond knowing the overall completion time for batch applications today; we also strive to determine the anticipated completion time for batch applications after the conclusion of a specific job.

Personas for batch job elapsed time prediction

Considering the prediction of z/OS batch execution end times, we can identify key factors that play significant roles in the process. These factors can be likened to Personas representing different responsibilities or influences on the accuracy of the prediction. Each persona represents a distinct role that contributes to the batch execution end prediction.

- IT operations personnel

  With the assistance of AI solutions, IT operations personnel can effortlessly monitor the health status of batch processing jobs and accurately predict their completion time. This AI-powered solution streamlines the monitoring and management of batch processing applications, ensuring efficient utilization of CPU resources while adhering to specified batch processing time windows.

  The capability to monitor and manage the execution status of batch processing applications using AI brings numerous benefits to IT operations personnel. By continuously monitoring the execution process, the solution can quickly identify any anomalies or deviations from expected behavior. This real-time insight allows personnel to promptly detect and address issues that may arise during batch processing execution, leading to improved efficiency and reduced downtime.

  Moreover, the AI-powered solution provides valuable insights into the overall health and performance of batch processing jobs. It can analyze various metrics, including resource consumption, execution times, and historical patterns, to identify potential areas for optimization. Armed with this information, IT operations personnel can make informed decisions to enhance the efficiency of batch processing operations.

- System administrator

  By utilizing the performance data from existing batch processing jobs, a z/OS system administrator will have the opportunity to gain valuable insights. Through thorough analysis and careful evaluation, he can identify areas for improvement within the architecture management. This strategic optimization aims to enhance the overall efficiency and effectiveness of the system.

  By examining the performance metrics and analyzing the behavior of the batch processing jobs, we can identify patterns, bottlenecks, and areas of potential optimization. These insights enable us to make informed decisions and implement changes to the architecture that align with the specific needs and goals of the organization.

  With the help of advanced analytics and AI-driven techniques, a system administrator can not only identify areas of improvement but also provide recommendations for architectural enhancements. These recommendations consider factors such as resource utilization, job dependencies, and system constraints. By incorporating these suggestions, he can optimize the architecture management process and unlock the full potential of the system. This proactive approach to architecture management allows system administrator to address any performance issues or inefficiencies promptly. By strategically optimizing the architecture based on analysis results and recommendations, he can ensure that the batch processing jobs run smoothly, meet deadlines, and maximize resource utilization.

  Ultimately, this data-driven and strategic approach empowers administrators and architects to make informed decisions, drive continuous improvement, and enhance the overall performance and efficiency of the batch processing system.

- Architect

  The z/OS architect has similar expectations to leverage the performance of existing batch processing jobs in order to strategically optimize architecture management based on analysis results and recommendations. However, in addition to the analysis, the architect requires a comprehensive visual reporting tool that provides a detailed overview of the daily batch processing executions for various applications.

This visual report should present a clear representation of the trends observed in the daily execution of batch processing jobs. It should include key metrics such as execution times, resource consumption, job dependencies, and other relevant factors. By visualizing this information, the architect can gain valuable insights into the performance patterns and identify areas for improvement.

With the aid of the visual report, the architect can easily identify any anomalies, bottlenecks, or inefficiencies in the batch processing workflows. The report's visual representation allows for a quick and intuitive understanding of the overall performance trends and the impact of specific applications on the system.

By having access to this comprehensive data and visual insights, the architect can make informed decisions and strategize optimizations to enhance the architecture management process. This may involve adjusting resource allocations, optimizing job scheduling, or refining dependencies between applications. The visual report serves as a powerful tool to support these strategic adjustments and optimizations.

Batch job elapsed time modeling building

The dynamic prediction of batch processing execution times allows us to anticipate the duration of each job accurately. This capability empowers enterprises to plan their workflows more effectively, allocate resources optimally, and streamline the overall data processing pipeline. As a result, potential bottlenecks and resource constraints can be proactively addressed, minimizing idle time and maximizing resource utilization.

To develop a model that effectively communicates with z/OS batch data of SMF30 records, we must carefully consider how this model can leverage the inherent nature and features of z/OS. Building a successful model that interacts seamlessly with z/OS batch data requires a thorough understanding of z/OS's unique characteristics and data structures. Specifically, we need to consider the way z/OS manages and processes batch data, as well as the specific information captured in the SMF30 records.

Define the prediction target

Predicting batch completion time before the batch begins is a crucial step in efficient batch processing. By leveraging historical data, analyzing job dependencies, and considering resource availability, we can estimate the expected time it will take for the entire batch to complete.

During this predictive phase, we can identify potential bottlenecks or areas of concern that may affect the batch's performance. This early insight allows us to proactively allocate resources and optimize job scheduling, ensuring smooth execution and meeting strict time constraints.

Additionally, predicting batch completion time in advance enables better planning and coordination across the organization. It facilitates effective resource allocation, ensuring that sufficient CPU power, memory, and storage are available for the batch's smooth execution. The accuracy of this prediction depends on the quality and relevance of historical data used for analysis. Continuous improvement of the prediction model through regular updates with fresh data enhances the precision of the estimates.

By being proactive in predicting batch completion time before it begins, we can maximize resource utilization, improve workflow efficiency, and boost overall productivity in the enterprise.

Another crucial step in efficient batch processing is to dynamically update the prediction during batch execution. Traditional batch processing systems often rely on static predictions made before the batch starts, if all tasks will run as initially estimated. However, real-world scenarios can be dynamic, with unpredictable factors that can impact the execution time of

individual jobs. For example, varying data volumes, system loads, or external events may affect the actual completion time of jobs within the batch.

To address the limitations of static predictions, modern batch processing systems are increasingly adopting dynamic prediction mechanisms. The key idea behind dynamic updates is to continuously monitor the progress of batch jobs during execution and adjust the predictions in real-time based on observed data:

- Real-time monitoring - Dynamic updates require real-time monitoring of job progress and resource utilization. This can be achieved through sophisticated monitoring tools and data analytics platforms that track key performance metrics, such as execution time, resource consumption, and job dependencies.
- Data-driven insights - The collected data during execution provides valuable insights into the actual performance of batch jobs. By analyzing this data, it becomes possible to identify patterns, trends, and potential bottlenecks that were not accounted for in the initial static prediction.
- Adaptive algorithms - Dynamic updates rely on adaptive algorithms that continuously learn from the incoming data. These algorithms can adjust their predictions based on new information, ensuring that the most accurate estimates are provided throughout the batch execution.

Hypothesis for model building

The hypotheses for model building are crucial when predicting batch application elapsed time. A hypothesis is a testable statement or assumption about the relationship between variables. It helps guide the model-building process and ensures that the model is developed with a clear objective in mind.

Some business-related information and metrics can affect the duration of batch processing jobs, such as daily transaction volume and the number of accounts. Batch processing plays a pivotal role in efficiently managing and processing large volumes of data in enterprises. However, the duration of batch processing jobs is not solely dependent on the technical aspects of the system. Instead, various business-related information and metrics can significantly influence the execution time of these jobs.

When it comes to batch processing, the performance and execution times of jobs are not isolated from the core business operations. Several business-related metrics directly impact how efficiently batch jobs are executed. Two critical factors that significantly affect the duration of batch processing jobs are:

- Daily transaction volume

  The daily transaction volume is a key business metric that can have a substantial effect on batch processing job duration. As transaction volume increases, the amount of data to be processed by batch jobs also rises. This can lead to longer execution times as the system needs to handle a larger data load.

  Moreover, varying transaction volumes on different days of the week or during peak hours can create fluctuations in batch processing job times. Understanding these patterns and trends in transaction volumes is essential for optimizing the scheduling and resource allocation for batch jobs.

- Number of accounts

  The number of accounts in an enterprise's database can directly influence the execution time of batch processing jobs. The more accounts that need processing, the longer the jobs will take to complete. This is particularly relevant for industries like banking and finance, where large customer bases result in a higher number of accounts to process.

In conclusion, when developing a predictive model for batch jobs, prioritizing these jobs based on their business significance becomes crucial for ensuring timely execution. High-priority jobs, such as critical financial transactions, must be given precedence to safeguard essential business operations from delays. Understanding the impact of each batch job on business outcomes allows for strategic prioritization, resulting in optimized resource allocation and improved overall system performance.

The successful implementation of a predictive model for batch job execution requires a careful assessment of the business impact of each job. By categorizing jobs based on their importance to core business operations, IT operations personnel can effectively manage the execution flow and allocate resources accordingly.

Another hypothesis for model building is during batch execution, the runtime of preceding jobs can influence subsequent jobs. The runtime of preceding jobs can significantly influence the execution of subsequent jobs, leading to potential bottlenecks and delays. In this section, we will explore the impact of job dependencies on job runtime during batch execution and delve into strategies to optimize performance and enhance overall system efficiency.

Batch processing involves executing a series of tasks in a predefined sequence, where the output of one job becomes the input for the next. These interrelated jobs form a complex web of dependencies, and their execution order can have a profound effect on the overall efficiency of the process.

Job dependencies can be broadly categorized into two types:

- Sequential dependencies - These dependencies occur when the execution of a job depends on the successful completion of the preceding job. The subsequent job can only commence once the preceding job has finished and produced the necessary data or results.
- Parallel dependencies - In this scenario, multiple jobs can run concurrently, and their execution is independent of each other. However, a subsequent job might depend on the collective output of the parallel jobs.

When considering job dependencies, the runtime of preceding jobs becomes a critical factor in the overall duration of the batch execution. If a preceding job takes longer than anticipated, it can lead to delays in the subsequent jobs, causing a cascading effect on the entire batch processing workflow.

Moreover, poor job sequencing can result in resource contention, where multiple jobs compete for limited resources simultaneously. This can lead to increased wait times and suboptimal resource utilization.

Job dependencies in batch processing create a sequence of tasks where the output of one job becomes the input for the next. The order in which jobs are executed can significantly impact the overall elapsed time of the batch processing workflow. Therefore, the predictive model must account for these dependencies to accurately estimate the time each job will take to complete.

Data engineering for batch job

As a z/OS-specific batch job elapsed time modeling process, the collection and utilization of historical training data are critical factors that require careful attention. Understanding the relevant data factors that need to be known is essential for building an effective predictive model. The following key data elements play a pivotal role in this process:

- Business data

  Business data refers to the information and records generated and collected by an organization during its operations and activities. The meaning and significance of

business data lie in its ability to support decision-making, improve operational efficiency, and gain a competitive edge.

The following key considerations should be taken for business data when analyzing and modeling batch job elapsed time:

- Daily online transaction volume

  It refers to the total number of transactions conducted within a specific online system or platform over the course of a single day. These transactions typically involve activities such as purchases, payments, transfers, or any other interactions made by users through the online platform.

  The “daily online transaction volume” can have a significant impact on the batch job elapsed time. Batch jobs often have dependencies, where the output of one job becomes the input for another. If the transaction volume is high, and the preceding jobs take longer to process the data, it can lead to delays in the subsequent jobs. This can cause a cascading effect, further elongating the elapsed time of the batch job workflow. The timing of the daily online transaction volume can also influence batch job elapsed time. For example, a spike in transaction volume during peak hours might lead to heavier batch processing, potentially causing job delays.

- Number of accounts

  Each account typically has associated data, such as customer details, transaction history, and preferences. With a larger number of accounts, the volume of data that needs to be processed by the batch jobs increases proportionally. This larger data set can lead to longer elapsed times for batch jobs as they process a larger volume of information.

  As the number of accounts grows, the demand for system resources, such as CPU, memory, and storage, also increases. The batch jobs may experience resource contention as they compete for these limited resources, potentially leading to longer processing times.

  The number of accounts can also influence the feasibility of parallel processing. If the system can effectively distribute the processing workload across multiple nodes or servers, it can mitigate the impact of many accounts on individual batch job elapsed times.

- Calendar-related information

  Calendar-related constraints may determine batch windows during which jobs can run. Limited batch windows may result in job delays, longer wait times, and overall elongated elapsed times if jobs cannot be executed immediately due to scheduling constraints.

  Calendar-related information can also impact the availability of personnel responsible for managing and overseeing batch job processing. Job scheduling, monitoring, and troubleshooting may need to align with personnel availability, influencing overall job execution times.

  Calendar-related information plays a crucial role in determining the elapsed time of z/OS batch jobs. By carefully considering business hours, time zones, holidays, end-of-month processing, seasonal workloads, and other calendar-related factors, enterprises can enhance batch processing efficiency, ensure timely execution of critical tasks, and align their data processing activities with business requirements.

- Batch feature data

  Batch feature data refers to the specific data elements or attributes that are relevant and used for the analysis, modeling, and prediction of batch processing jobs. These features

play a critical role in building machine learning models to understand the behavior and performance of batch applications.

When dealing with batch feature data, several important aspects should be considered:

– Job list (names of all jobs belonging to the batch)

  The job list provides essential information about the order in which jobs are scheduled to run. Batch jobs may have dependencies, where the output of one job serves as input to another. Understanding the job list enables the predictive model to account for job dependencies and accurately predict the elapsed time of each job and the overall batch workflow.

  The job list can also indicate the availability of batch windows during which jobs can be executed. These time constraints may influence the scheduling and resource allocation decisions, impacting the elapsed time prediction.

– Start job and end job of the batch

  The “Start job and end job of the batch” have a significant impact on the z/OS batch job elapsed time prediction. These two key points help determine the boundaries of the batch job execution, providing essential information for accurate time prediction.

  The “Start job” marks the beginning of the batch job workflow, while the “End job” indicates its completion. Knowing these points allows the predictive model to focus solely on the relevant time range for making predictions. By considering only the jobs between the start and end points, the model can efficiently analyze the required data and optimize predictions for the specific batch execution.

  Knowing the start and end jobs helps in understanding the batch window constraints within which the jobs must be completed. This information influences job scheduling decisions, ensuring that jobs are executed within the specified time frame and potentially minimizing delays.

  The order in which jobs run in the batch job workflow is essential for the prediction. The start job and end job provide context for job sequencing, enabling the model to understand the dependencies and relationships among jobs. Accurate job sequencing contributes to precise elapsed time predictions.

In a real z/OS batch job, several performance metric data can be collected and analyzed to gain insights into its performance. Some of the performance metrics that can be obtained are:

- Elapsed time
- CPU time
- I/O numbers
- Start and end time
- One year of historical data used for analysis and modeling
- SMF30 performance data captured on z/OS, subtype=5

Significance of SMF30 records in batch job elapsed time prediction modeling

Let us emphasize the significance of SMF30 records and why they are essential for batch job elapsed time prediction modeling.

System Management Facility 30 (SMF30) records are critical components of z/OS that capture essential performance data related to batch job accounting and monitoring. These records provide valuable insights into the execution of batch jobs, offering a wealth of information, including job identification details, resource utilization metrics, job start and end times, elapsed time, job step information, and execution statistics. For batch job elapsed time prediction modeling, the inclusion of SMF30 records is crucial for several reasons.

SMF30 records also offer a comprehensive set of performance metrics related to batch job execution. These metrics encompass CPU time, I/O operations, memory usage, and other crucial indicators that directly impact the elapsed time of batch jobs. By integrating these metrics into the modeling process, the predictive model gains a holistic view of job performance.

By leveraging SMF30 records, the prediction model becomes data-driven, relying on actual performance metrics rather than assumptions or generalizations. Data-driven decision-making empowers organizations to make informed choices, optimize resource utilization, and enhance the efficiency of batch job processing.

Now, let us delve into an example of SMF30 record data, shown in Example 5-2. We assume that the SMF30 data has already been loaded and processed into the desired format for analysis. The time range covered in the analysis spans from November 1, 2019, to October 30, 2020. Within this period, the job list range includes jobs with identifiers from `APP001` to `APP050` for each day. This ensures a comprehensive overview of batch job performance and enables a detailed examination of job execution patterns and trends throughout the specified time frame.

Example 5-2 Elapsed time batch execution time

```
smf30_file_path=r'data/Sample_SMF.csv'
job_list_file_path=r'data/all_joblist.csv'

APP_NAME = "AP"
start_jobs = ['APP001']
cutoff_jobs = ['APP050']
```

Normalized data for SMF30 metrics

In Figure 5-7 on page 98, we have created a chart of the daily sums for JOB_DURATION_SECS, CPU_TOTAL_SECONDS, JOB_CPU_SECONDS, ELAPSED_TIME_SECONDS, and JOB_IO_COUNT during the specified period between November 1, 2019, and October 30, 2020. Among these metrics, ELAPSED_TIME_SECONDS holds particular significance for our modeling efforts. This metric indicates the duration of batch job execution for each day throughout the specified time frame.

By paying close attention to ELAPSED_TIME_SECONDS, we can gain crucial insights into how long the batch jobs run daily during the observed phase. This information is invaluable for building predictive models and optimizing batch job performance. However, there are other related metrics that can also provide insights into the batch job's performance, such as CPU time, I/O count, and more.

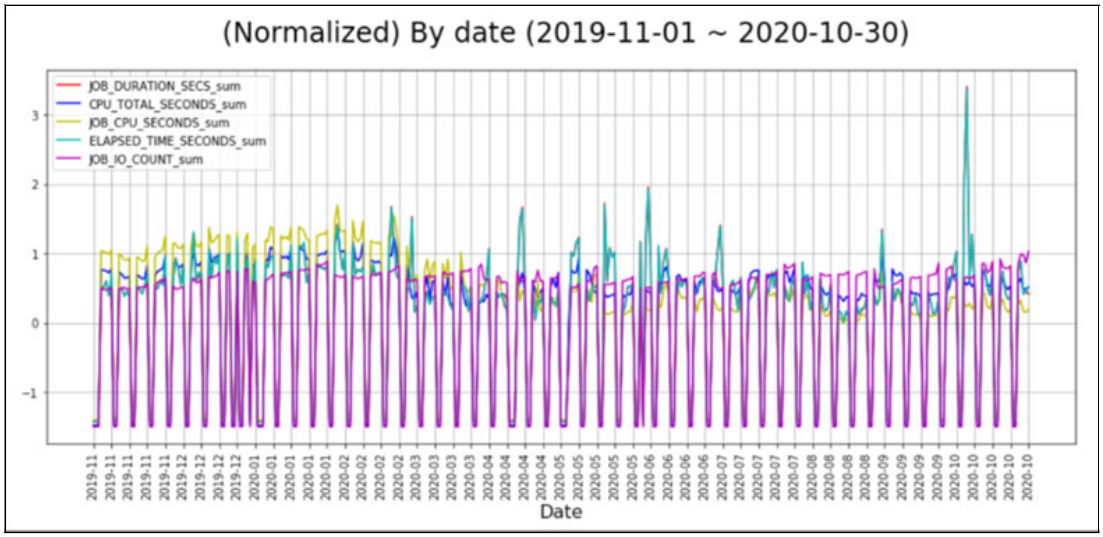


Figure 5-7 Normalized SMF30 data for batch application

In Figure 5-8, we present a detailed chart showcasing elapsed time from various aspects, namely ELAPSED_TIME_SECONDS_EXT, ELAPSED_TIME_SECONDS_INT, and ELAPSED_TIME_SECONDS_SUM. These metrics provide comprehensive insights into different aspects of the batch job execution time, enabling a deeper understanding of its performance.

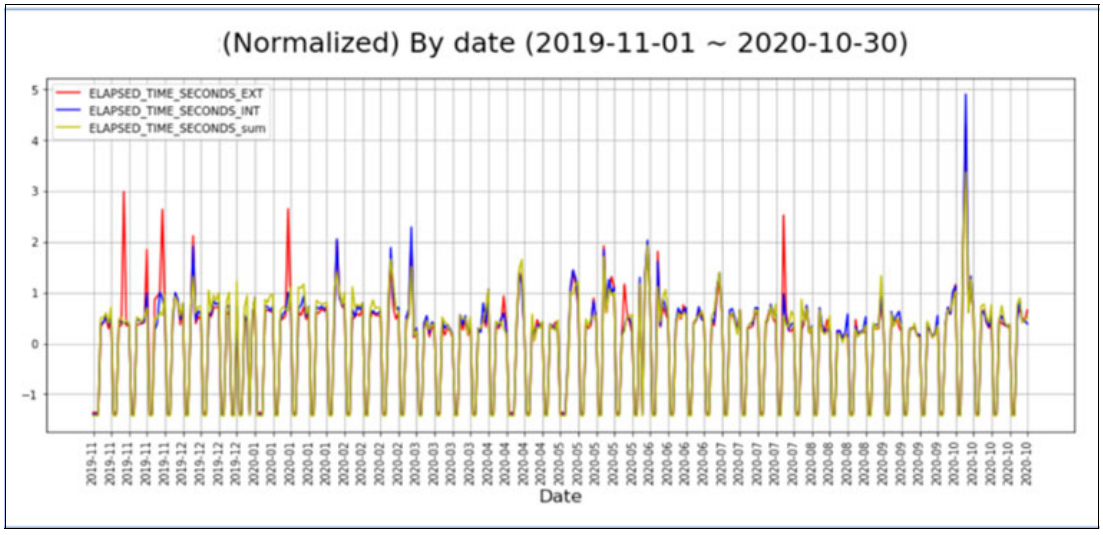


Figure 5-8 Normalized SMF30 Elapsed time for batch application

Let us further explore the elapsed time example, using Figure 5-9 on page 99 to gain a better understanding of the following metrics: ELAPSED_TIME_EXT, ELAPSED_TIME_INT, and ELAPSED_TIME_SUM.

Job Name	Start Time	End Time	Elapsed Time (min)
Job1	1:00	1:02	2
Job2	1:00	1:03	3
Job3	1:10	1:11	1

Figure 5-9 Batch application with sample jobs elapsed time

We can use the formulas shown in Example 5-3 to calculate the three key, before mentioned metrics.

Example 5-3 Elapsed time metrics

```
ELAPSED_TIME_EXT: 11 minutes. ## the model prediction target.
Elapsed Time with External Application Waiting Time=
 max(End  Time)-min(Start Time)=1:11-1:00=11 minutes.
*******************************************************
ELAPSED_TIME_INT: 4 minutes.
Elapsed Time within Internal Application=
   sum(continuous elapsed time)=(1:03-1:00)+(1:11-1:10)
                                =3+1=4  minutes.
*******************************************************
ELAPSED_TIME_SUM: 6 minutes
Elapsed Time of all individual jobs=
    sum(every job's elapsed time)=2+3+1=6 minutes.
```

Now that we understand the meaning of elapsed time, let us revisit Figure 5-10 on page 100. Upon closer examination, we can observe several red peaks, indicating instances where the batch transaction's elapsed time significantly exceeds the expected time it should take for internal application execution. These red peaks are essential indicators of potential performance issues or inefficiencies within the batch processing, warranting further investigation and optimization.

Next, we explore the importance of investigating the monthly elapsed time for September 2020 and daily elapsed time for September 15, 2020, batch execution.

If there are concerns about potential performance issues for a particular month, you can refer to the monthly elapsed time for batch job report, shown in Figure 5-10 on page 100, to investigate the elapsed time. This report provides valuable insights into the time range of batch job execution, in our example, for September 2020.

Pay close attention to the comparison between ELAPSED_MINUTES_EXT and ELAPSED_MINUTES_INT. If there is a substantial gap between these metrics, it indicates a significant difference between the externally observed elapsed time and the internally expected elapsed time. Such discrepancies might be indicative of performance anomalies that warrant further investigation and optimization.

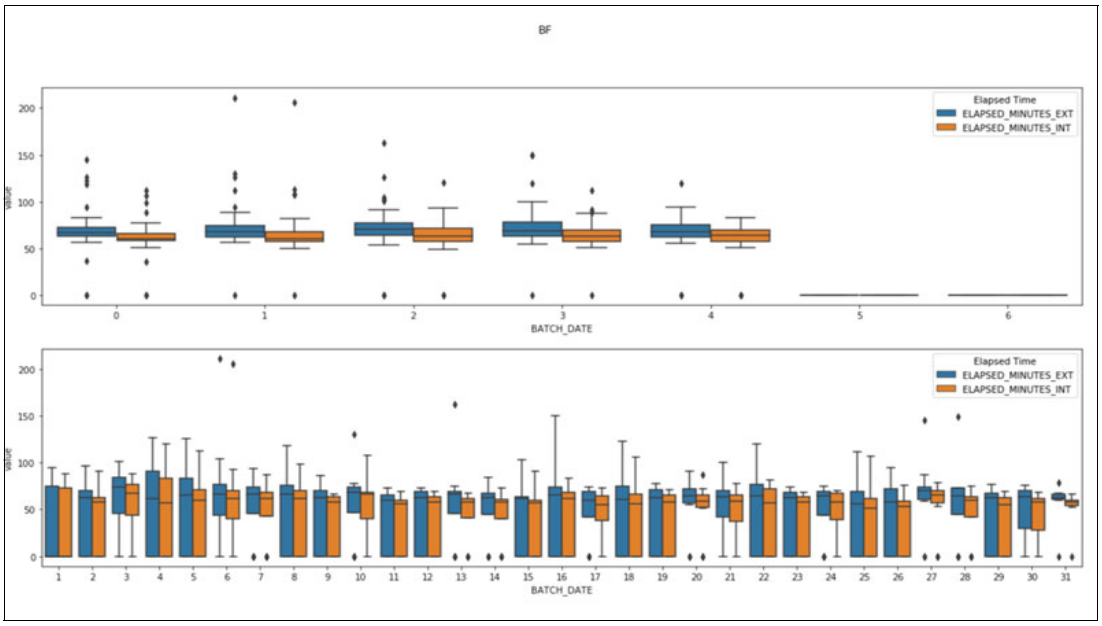


Figure 5-10 Monthly elapsed time for September 2020

Additionally, Figure 5-11 on page 101 provides a comprehensive overview of all batch jobs executed on a specific day. The figure displays the duration time for each job within the day, represented by progress bars that indicate the job execution time. This visual representation allows for a quick and efficient review of job performance throughout the day, enabling easy identification of any jobs with longer execution times or potential inefficiencies. By leveraging this graphical representation, users can gain valuable insights into job execution patterns, optimize resource allocation, and streamline batch processing to enhance overall efficiency. This graphical approach simplifies the interpretation of complex data, making it easier to make informed decisions, streamline job processing, and optimize system performance.

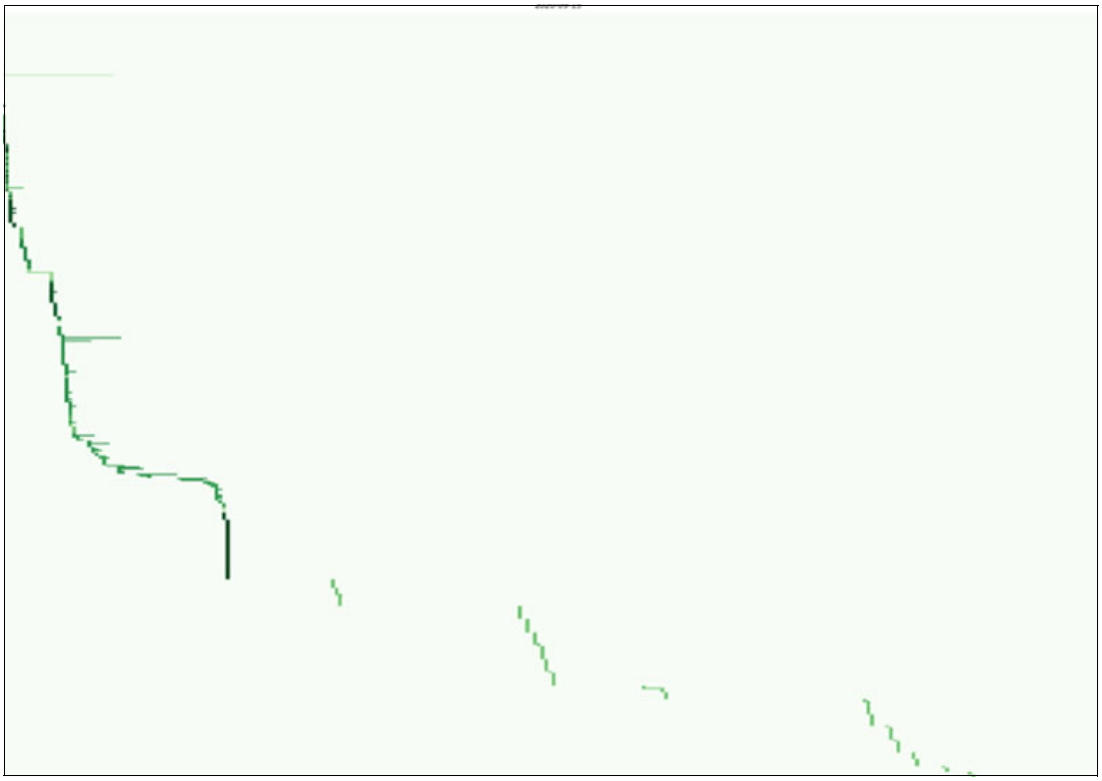


Figure 5-11 Elapsed time for 2020-09-15

Build predictive models in a stepwise manner

Before delving into model building, it is crucial to understand the following two key concepts: Influential jobs and Milestone jobs.

Influential Jobs refer to specific batch jobs that have a significant impact on the target variable in our predictive model. These jobs play a pivotal role in determining the outcome of the model's predictions. By identifying influential jobs, we can focus our analysis on the most critical factors that contribute to the performance of the batch processing system. Analyzing correlations between these influential jobs and the target variable helps us gain valuable insights into their importance and influence on overall batch job performance.

The essential characteristics of an influential job include:

- These elements must be consistently present in each daily batch application.
- Influential jobs are strongly correlated with batch execution time statistically and they demonstrate a strong correlation with the target variable.
- Influential jobs can be identified by the correlations between predictors and target.
- Incorporate jobs deemed important from a business logic perspective based on expert experience as influential jobs.

Conduct correlation analysis between each job and the target variable (e.g., batch execution time). Calculate the correlation coefficient to measure the strength of the relationship between each job and the target. In this article, we assume that the data scientist has conducted a thorough analysis about all jobs from the data set, marking the jobs that have been identified as influential from their experience.

Milestone Jobs refer to a sequence of Influential jobs in the batch application path. It represents critical checkpoints or key events within the batch processing system. These jobs

mark significant progress or completion of essential steps during batch job execution. Understanding the timing and performance of milestone jobs allows us to track the progress of batch job processing and evaluate their individual impact on elapsed time. By leveraging this information, we can optimize batch job scheduling and resource allocation, leading to enhanced system efficiency. In Figure 5-12, the milestone jobs are a sequence of influential jobs in a batch application.

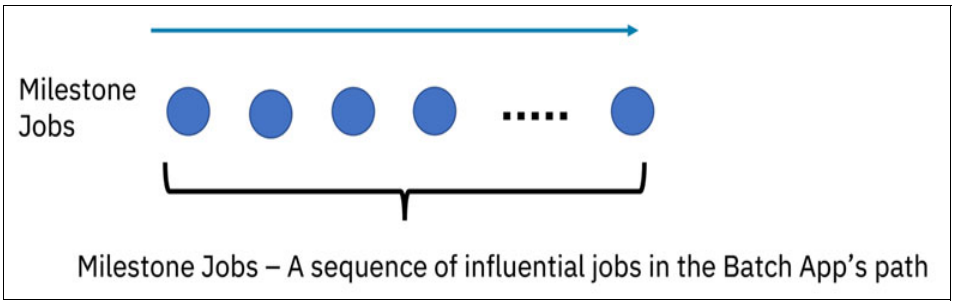


Figure 5-12 Milestone jobs - a sequence of influential jobs

In this solution, a critical task for model building is to identify the correct milestone jobs. Influential jobs within these milestones play a pivotal role as significant checkpoints during the execution of the batch application. These influential jobs tend to adhere to a specific order of execution. Identifying the appropriate milestone jobs is crucial as they provide essential reference points for monitoring the progress and dependencies within the batch application. By understanding the sequential order of influential jobs within these milestones, data engineering and model building can optimize the flow of batch job execution, ensuring efficient resource allocation and streamline performance.

Constructing the influential jobs chain and identifying the milestone jobs

Next, let us explore the process of constructing the influential jobs chain and identifying the milestone jobs with this sequence of influential jobs using a sequence pattern analysis on influential jobs with Spark PrefixSpan, a projection-based algorithm.

PrefixSpan works by recursively searching for frequent sequential patterns, extending prefix patterns in a depth-first manner. It uses a depth-first search strategy to efficiently explore the search space of potential sequential patterns. The algorithm utilizes a projection technique to avoid redundant computations, making it highly scalable for large data sets.

Spark PrefixSpan is an implementation of the PrefixSpan algorithm specifically designed to leverage the distributed computing capabilities of Apache Spark. By leveraging the Spark PrefixSpan algorithm, we streamline the process of identifying influential jobs and milestone jobs, enabling us to optimize job scheduling, resource allocation, and overall system efficiency.

Please see the handling for influential jobs sequence with PrefixSpan:

1. Split the entire time frame into several time windows.

   This step involves dividing the data set representing the batch application's execution into smaller intervals, or time windows. These time windows help organize the data and facilitate the analysis of influential jobs within specific periods.

2. Process the data from the last time window.

   Process the data from the last time window and then proceed to the previous window. The latter window should encompass more critical z/OS batch transactions. As we progress from the last window to the previous window, the data is likely to include batch transactions that hold greater significance or importance.

3. Manually define several key time points in the last window.

   After handling the last time window, we iterate the same processing with the previous time window. This iterative approach allows us to analyze the influential job sequence and milestone jobs for each time window in a stepwise manner, effectively capturing the dynamic behavior of the z/OS batch application over time. See Figure 5-13 for an example of key jobs' sequence of influential jobs in a time window.

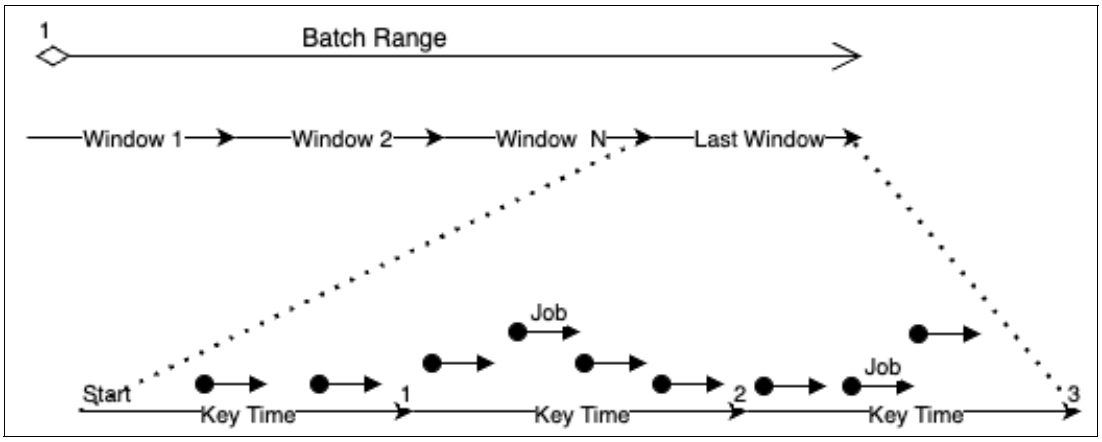


Figure 5-13 PrefixSpan for influential jobs sequence

4. Compute the score for each job at every key time using PrefixSpan.

   With the PrefixSpan algorithm, we calculate a score that corresponds to the time gap between the job's completion time and the key time. This score provides valuable information about how each job aligns with a key time, indicating whether it was completed before or after the critical moment. The score computing method is represented by the formula provided in Example 5-4.

Example 5-4 PrefixSpan algorithm to calculate individual score for each job

```
Score = log(execution_time)*
                    abs(completion_time -  key_time)/time window
```

5. Compute the average score for each job at every key time.

   After obtaining individual scores for each job with respect to the key times using the previously mentioned formula, we proceed to compute the average score for each job across all key times.

   To calculate the average score for a specific job, we sum up the scores of that job at each key time and then divide the sum by the total number of key times. This process allows us to derive an overall measure of the job's temporal alignment with the critical events throughout the batch application's execution.

6. Select the job with the minimum score across all jobs.

   From the computed average scores for each job, we identify the job that has the minimum score.

   This job represents the most critical contributor to the timely completion of the batch application's execution, as it consistently aligns well with the key time.

7. Add all key jobs in the previous time window as dependencies.

   Once the job with the minimum score is identified, we consider all the key jobs present in the previous time window. These key jobs serve as essential reference points for tracking

progress and dependencies within the batch application. The key jobs are also called influential jobs.

Now we establish a meaningful chain of influential jobs with their respective dependencies, which represents a significant sequence of batch job execution. In other words, milestone jobs are identified now. As a result, such identification of milestone jobs is a crucial step towards achieving better batch processing outcomes and meeting business objectives efficiently.

Model building in a stepwise manner

A stepwise manner in machine learning offers numerous benefits, including improved model performance, resource efficiency, interpretability, and flexibility. By breaking down the model-building process into manageable steps, it helps ensure a more effective and streamlined development process, leading to accurate and reliable machine learning models. IBM z/OS batch applications can be highly complex, involving numerous interconnected jobs with dependencies. A stepwise approach allows us to break down this complexity into smaller, more manageable tasks, making it easier to understand, develop, and optimize the application.

In Figure 5-14, we have established multiple milestone phases, each representing a distinct segment of the z/OS batch application's execution. We recognize that within each milestone phase, there exist a set of influential jobs that significantly impact the overall performance of the batch application. To capture these effects accurately, we propose training a separate predictive model for each milestone phase.

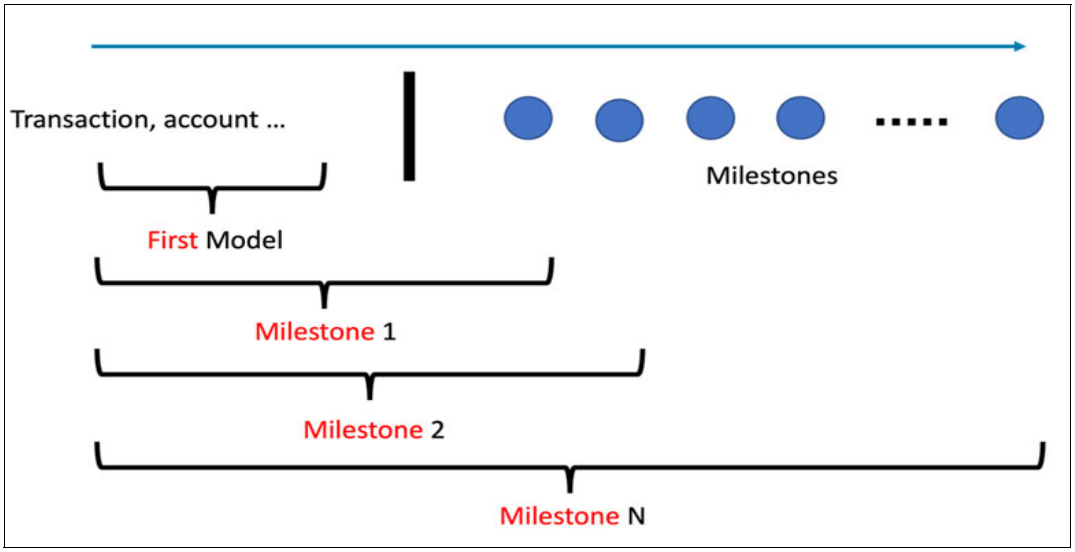


Figure 5-14 Milestones for a batch application

By training individual models for each milestone phase, we can obtain a more granular understanding of the influential jobs' execution patterns within that specific time frame. These models will reflect the dynamic nature of the influential jobs and their impact on the batch application's elapsed time during each milestone phase.

This stepwise approach ensures that the predictive models are tailored to capture the unique characteristics and dependencies of influential jobs within each milestone phase. It allows us to optimize the models for specific time frames, which can lead to more accurate predictions and better performance throughout the batch application's execution.

Additionally, the milestone modeling process can be seamlessly integrated with users' job logs and daily z/OS batch transaction data. As shown in Figure 5-15 on page 105, we employ

the job log data to train predictive models for each milestone. These models are specifically designed to capture the elapsed time patterns of influential jobs within the milestone phases.

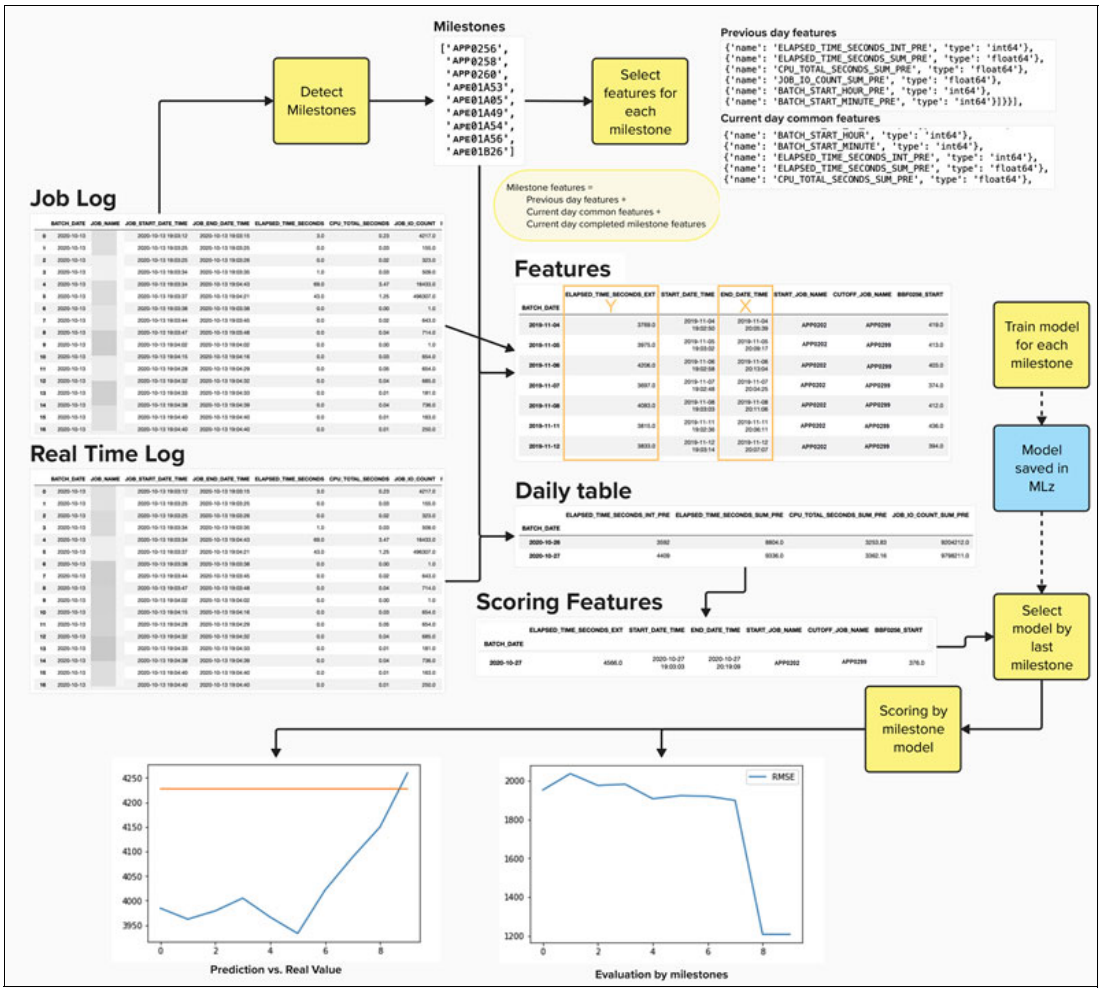


Figure 5-15 Batch jobs elapsed time prediction process

After training milestone models, we leverage real-time job log data to score the elapsed time for the ongoing batch application execution. By using milestone models to predict elapsed time, we obtain accurate and up-to-date estimations of job completion times during the execution process. Integrating the job log data and z/OS batch transaction data with the milestone modeling enhances the precision and reliability of the predictive models. Job logs provide detailed information about job execution, errors, and resource utilization, enabling us to fine-tune the models based on actual performance data.

By scoring elapsed time in real-time with the milestone models, we can monitor the progress of the batch application and identify any deviations from expected job completion times. This allows us to take prompt actions in case of delays or issues, ensuring that the batch application remains on track and meets its performance goals.

5.4.2 Applying batch job elapsed time model with MLz for application integration

In this section, we will explore the practical application of the batch job elapsed time model using the power of MLz for seamless application integration. After successfully training and

saving our predictive models, the next step is to deploy and implement them in real-world scenarios.

With the help of MLz model deployment and scoring capabilities, we can easily integrate the predictive models into our existing z/OS batch application workflows. This integration allows us to leverage the predictive insights from the models in real-time, enhancing the efficiency and performance of our batch processing.

As the z/OS batch application evolves, the milestone models undergo iterative updates and versioning, which are conveniently managed within the MLz model metadata repository. This versioning feature allows us to keep track of different iterations and improvements made to the models over time. It ensures that we have access to previous versions of the models for comparison and reference, as needed.

Once the milestone models are ready for real-time log prediction, they can be deployed or scored using the MLz scoring engine. This integration allows us to utilize the models' predictive capabilities during the actual execution of z/OS batch applications. By scoring the real-time logs with these milestone models, we continuously receive updated predictions of influential job elapsed times, empowering us with data-driven insights to make adaptive decisions and optimize the batch processing performance.

Batch job model with MLz model management

In Figure 5-16, we observe the generation of several milestone models, each capturing a specific phase of z/OS batch application execution. These milestone models play a critical role in predicting the elapsed time of influential jobs within their respective phases. To ensure seamless application integration, these models need to be persisted for future prediction.

By saving the trained models into the MLz metadata repository, we gain several advantages. One key advantage is model versioning, which allows us to keep track of different model iterations and updates over time. This versioning feature ensures that we can always access and refer to previous versions of the model, if needed. It provides a complete history of model development and helps us understand how the model has evolved over different stages of refinement.

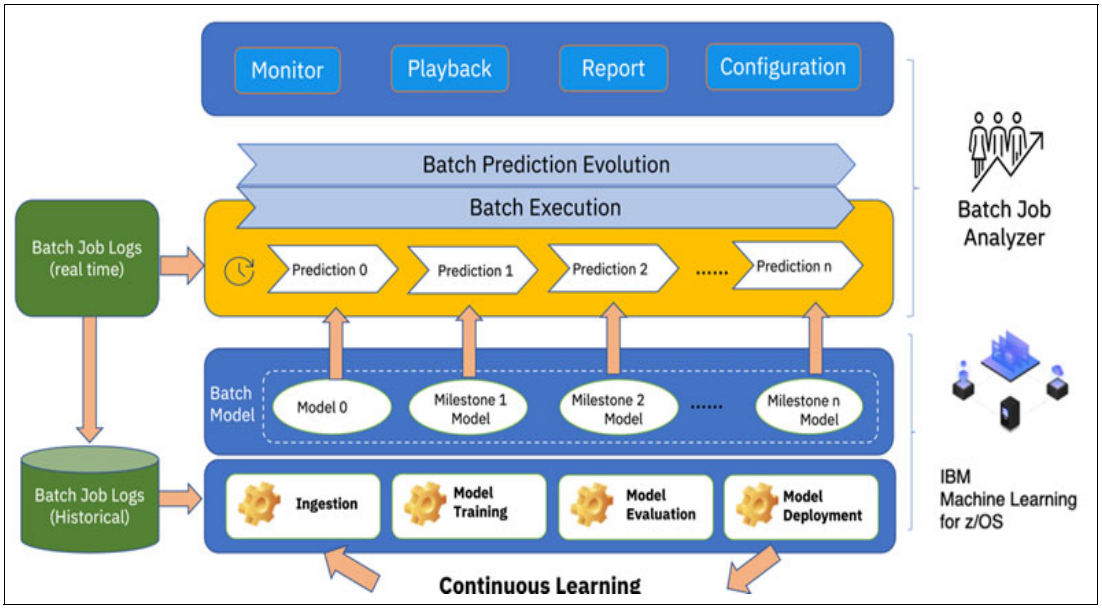


Figure 5-16 Batch jobs elapsed time use case integration with MLz

Furthermore, saving the models in the MLz metadata repository facilitates easy reloading and access for scoring purposes. The MLz scoring engine can seamlessly retrieve the stored models, making it convenient to perform real-time scoring for new batch application executions. This capability allows us to make use of the latest model insights for accurate and up-to-date predictions, contributing to the overall efficiency and effectiveness of the z/OS batch applications.

In MLz, the process of saving and managing the trained models is simple and straightforward. Once you have completed the model training, you can easily specify the model, along with the training data and target, in the MLz interface. Upon doing so, you will find the model conveniently listed in the MLz model repository. Example 5-5 outlines the process of saving a model using the MLz model management API.

Example 5-5 Batch model with MLz model management API example

```
def save_model(self, model, model_name, train_x, train_y):
  ml_client = MLRepositoryClient(self.core_service_url)
  ml_client.authorize_with_token(self.access_token)
  props1 = MetaProps({
           MetaNames.AUTHOR_NAME:self.MLz_info.user_name,
           MetaNames.AUTHOR_EMAIL: "batchprediction@cn.ibm.com"})
  input_artifact = MLRepositoryArtifact(model, name=model_name,
                   meta_props=props1,
                   training_data=train_x.head(),
                   training_target=train_y)
  saved_artifact =  ml_client.models.save(artifact=input_artifact)
  dict_meta = saved_artifact.meta.get()
  model_version_url = dict_meta['modelVersionUrl']
  print("model_version_url = " + model_version_url)
```

Batch job model prediction with MLz scoring service

The process of batch job model prediction with the MLz scoring engine involves leveraging the predictive models we have developed for batch job elapsed time.

The batch job model prediction with the MLz scoring engine allows us to harness the power of predictive analytics in real-time. By deploying and scoring the predictive models during z/OS batch application execution, we gain valuable insights for adaptive decision-making and performance optimization.

In Figure 5-17 on page 109, we can see that multiple scoring requests are sent based on each milestone, allowing us to dynamically review the predicted elapsed response time for the batch execution of specific milestone jobs. This real-time prediction and review process plays a crucial role in enhancing the efficiency and performance of the z/OS batch applications.

As the z/OS batch application progresses through different milestone phases, the milestone models are utilized to continuously predict the elapsed time of influential jobs within each phase. These predictions are generated dynamically as the batch application executes, providing valuable insights into the expected performance of critical jobs.

The scoring requests sent to the MLz scoring engine enables us to access the latest predictive capabilities and obtain real-time updates on the elapsed response time for each milestone job. By comparing these predictions with the actual performance during the execution, we can identify any discrepancies or potential performance issues in real-time.

Provided in Example 5-6 on page 108 is a pseudo-code sample demonstrating the MLz scoring process for batch job elapsed time prediction. When integrating MLz into your user

application, you can conveniently package the scoring endpoint request. The primary objective is to invoke the MLz scoring engine, which is responsible for the inference process. This seamless integration allows your application to leverage the power of MLz and obtain real-time predictions from the deployed machine learning models. By making scoring requests to the MLz scoring engine, your application can quickly process data and receive valuable insights, enabling you to make informed decisions based on the model's predictions.

Example 5-6 Batch model with MLz model scoring API example

```
test_data =
joblog_df[(joblog_df[batch_config.batch_date_col].isin(['2020-10-26','2020-10-27']
))]

score_url =
'https://MLz_scoring_ip:MLz_scoring_port/iml/v2/scoring/online/model_deployment_id
'
MLzClient = MLz_Scoring_Client(scoring_endpoint=score_url)
res = MLzClient.scoring(t_w_df)
```

Again, looking at Figure 5-17 on page 109, we can observe the batch job prediction results for a specific time point, which is around 20:17. The progress bar in the first line indicates a green status, suggesting that the predictive finish time for the entire batch application is expected to fall within the normal time range based on the current milestone job `APE01A05` point.

Although the real `APE01A41` job is marked as red and appears to exceed the normal expected endpoint, the overall batch application does not seem to be significantly impacted. The green status of the progress bar indicates that the batch execution is progressing smoothly and is expected to complete within the desired time-frame.

The predictive capabilities of the milestone models allow us to anticipate the expected elapsed times for influential jobs at different time points within the batch execution. By comparing the real-time performance of specific jobs, such as `APE01A41` in this case, with the predictions, we can quickly identify any deviations or potential performance issues.

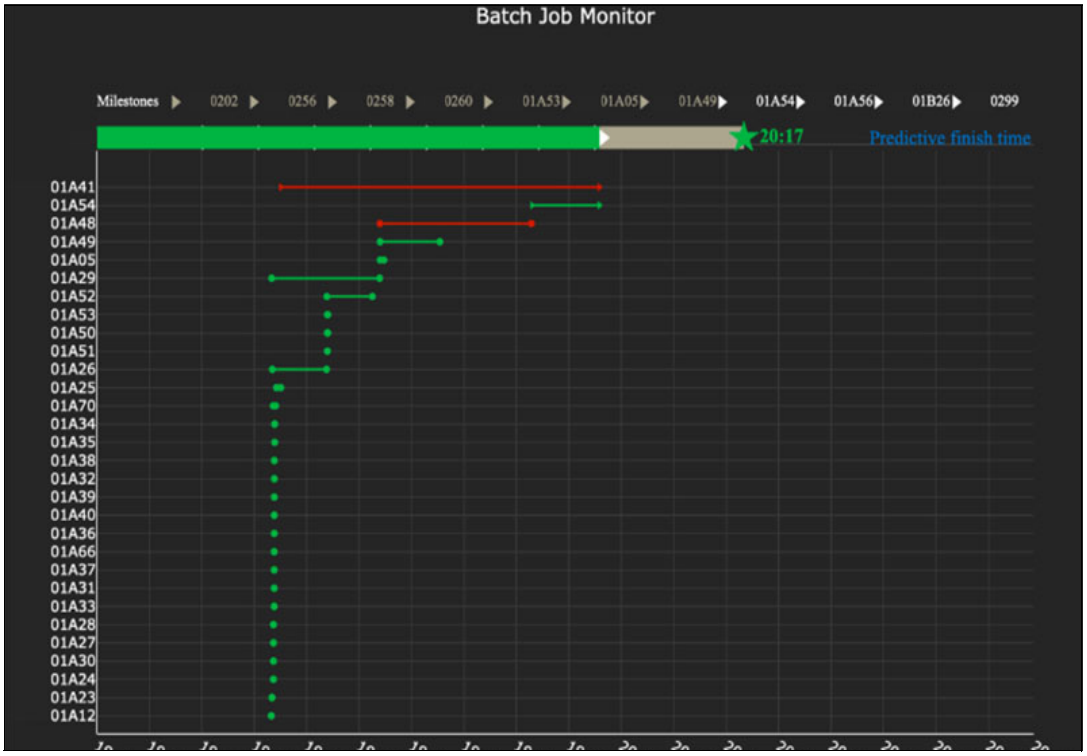


Figure 5-17 Batch job elapsed time prediction for normal case

In Figure 5-18 on page 110, we can observe the batch job prediction results for a specific time point, which is around 21:33. The progress bar in the first line is marked with a red status, indicating that the predictive finish time for the entire batch application is likely to be outside the normal time range based on the current milestone job `APP0260` point.

The red status of the progress bar raises a concern that the batch execution may not meet the desired completion time. The milestone models have predicted a longer elapsed time for influential jobs within the current phase, which suggests potential performance issues impacting the overall batch application.

As we focus on the real `APE01A41` job, it is also marked with a red status, further confirming that this specific job is experiencing a performance bottleneck or delay that is affecting the overall batch processing. The deviation from the predicted elapsed time for `APE01A41` indicates that there may be a need to investigate and address the root cause of this performance issue promptly.

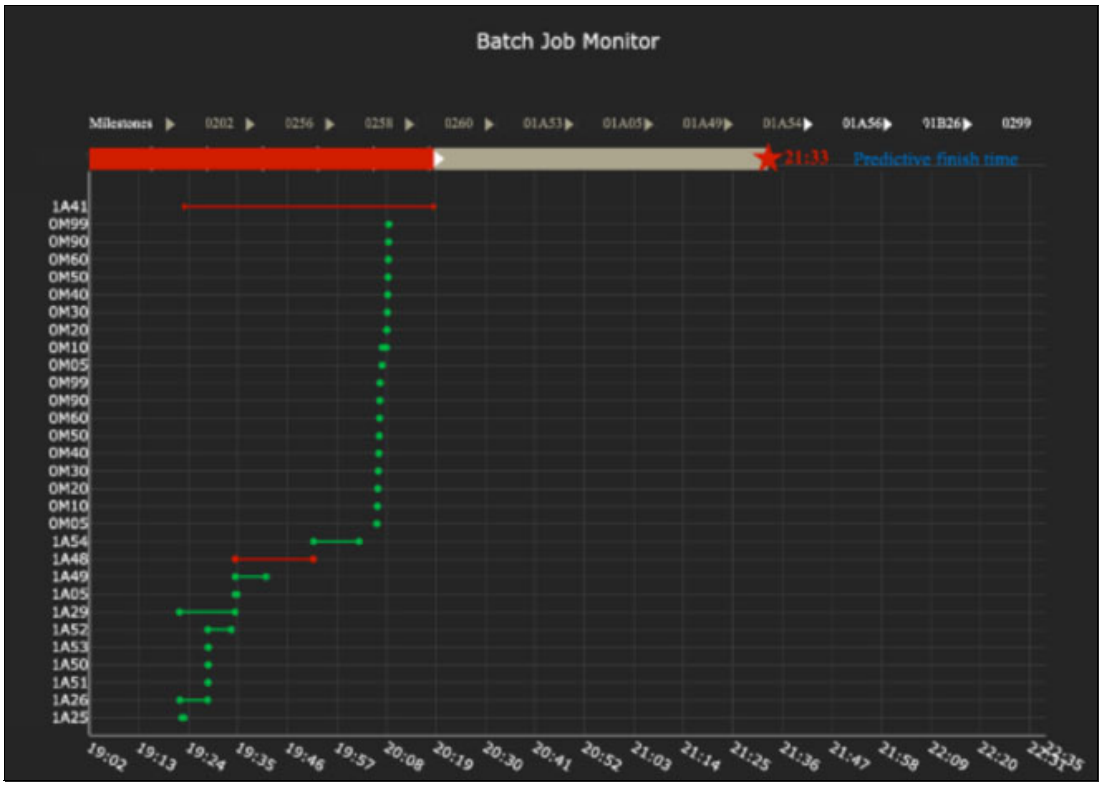


Figure 5-18 Batch job elapsed time prediction for error case

In conclusion, the real-time review of batch job predictions provided by the milestone models empowers us with valuable insights into the performance of individual jobs and the overall batch application. By promptly identifying and addressing potential issues, we can take proactive measures to optimize the batch processing efficiency and meet critical business deadlines.

The MLz Repository Service and Scoring Service are invaluable components for seamlessly integrating a batch prediction solution into your z/OS environment. These services enable you to efficiently manage and utilize predictive models, ensuring optimal performance and timely completion of critical batch jobs. By leveraging the MLz Repository Service and Scoring Service, you can fully integrate predictive analytics into your z/OS batch processing environment. These services enable you to deploy, manage, and score predictive models effectively, providing valuable insights into batch job elapsed times and empowering you to optimize the performance of critical batch applications.

5.5 Best practice on model deployment and inference

In today's dynamic world of machine learning and artificial intelligence, the deployment and inference phase of your models is where theory meets the real world. It is a critical juncture that can significantly impact the performance, scalability, and reliability of your applications.

5.5.1 Optimizing ML model integration and deployment for business decisions

When integrating and deploying machine learning models, a significant challenge lies in bridging the gap between the model training environment and the production setting,

particularly when AI is employed to optimize business decisions. Model selection is crucial, aiming for a well-rounded compromise between accuracy and inference speed. Various machine learning and deep learning frameworks like TensorFlow, PyTorch, and scikit-learn are accessible for model development, while production environments encompass diverse programming languages and platforms.

Traditional sectors such as banking and health care rely on distinct programming languages and systems, making rebuilding the entire infrastructure impractical. Instead, a recommended approach involves converting framework-specific models into exchangeable formats to ensure interoperability across training and production contexts.

Predictive Model Markup Language (PMML) and Open Neural Network Exchange (ONNX) are two widely adopted formats. PMML, an XML-based format, supports diverse machine learning frameworks like scikit-learn and SAS. ONNX, a versatile standard format, excels at representing deep learning models and enables model portability across various frameworks and platforms.

Popular deep learning frameworks, including TensorFlow and PyTorch, support direct exporting of models to ONNX format, facilitating seamless deployment and decoupling model development from deployment. Enhancing model inference speed involves techniques like model quantization, pruning, and compression to reduce model size. ONNX further streamlines inferencing in a unified environment, empowering data scientists to leverage their skills in both model development and deployment processes.

Nevertheless, the accuracy of a model does not hinge solely on the number of iterations during the training process. Equally crucial is the caliber and volume of the data employed for model training. Poor training data can result in the misinterpretation of trends, biases, an inability to identify novel patterns, inexplicable model outputs, and more. The training data set must exhibit sufficient volume while also being meticulously refined, accurate, consistent, and imbued with meaningful information.

5.5.2 Choose the right inferencing interface for your use case

The MLz scoring server offers a range of user-friendly programming interfaces specially designed for developers working on the z/OS platform. These interfaces include options like REST, Java, and CICS, as well as WOLA. When it comes to selecting the most suitable inferencing interface for your specific needs, there are certain factors to consider. The decision should be based on your main priorities and concerns. For instance, if you value quick response times and high throughputs, opting for the CICS or WOLA transactional inferencing interface would be beneficial.

Additionally, your choice depends on the nature of your application. If your application is built using Python or Java, the REST APIs are a convenient option. However, if your application is developed using an IBM Z native language, such as COBOL, using REST APIs might not be the best fit. Instead, choosing CICS or WOLA interface will be a better option.

Security requirements also play a role in your decision-making. If your IBM Z system has specific security needs, using REST APIs might introduce network-related risks that you need to be cautious about.

In essence, making the right choice among these inferencing interfaces involves a careful evaluation of your priorities, application type, performance requirements, and security concerns, ensuring that you select an interface that aligns seamlessly with your overall objectives and technical landscape.

6

Streamlining AI from models to applications using machine learning operations

In this chapter, we delve into the art and science of seamlessly transitioning from machine learning models to real-world applications. Machine learning operations (MLOps) is the catalyst that transforms theoretical brilliance into operational brilliance, marrying the realms of data science, engineering, and deployment. Get ready to embark on a journey where efficiency, scalability, and ethical considerations converge to not only enhance the capabilities of your models but to redefine how AI makes a tangible impact on the world. This chapter is your guide to navigating the intricate path from models to applications, where MLOps is the compass guiding you towards operational excellence in the realm of artificial intelligence.

6.1 Understanding MLOps

Breaking it down, machine learning operations (MLOps) represents streamlined and automated machine learning, data engineering, and DevOps practices throughout the entire lifecycle of machine learning (ML) models.

One of the key value propositions of AI on IBM Z is having your data and core applications collocated in one place. In the previous sections, we highlighted how the software development process has evolved with time to keep up with the increasing changes in hardware and compute capacity (starting with waterfall - agile-DevOps).

We also defined Machine learning as not a standalone piece of code or block that can exist without the training data, the incoming stream of new data, or the application that the ML model will be co-located with and is inferencing. With new data, training, and inferencing will continue to evolve and change. These changes will impact the quality and accuracy of the model and ML teams will actively monitor the model for indicators that will confirm model degradation. Those models must be updated with possibly new training and thus a new version of the model will have to be deployed.

In a workflow where the development process is agile, the process can break quite easily without a robust and scalable way to integrate the changes in the model with the rest of the components. Development operations or DevOps alleviates that by enabling IT teams apply a systematic, recorded, and repeatable process to streamline and continuously monitor the integration and deployment of their application changes with the overall infrastructure. At an enterprise level where without a scalable process like this, there can be severe business and technical impacts on the teams involved and the overall business.

MLOps combines machine learning with software development and overall DevOps to deliver machine learning models into production at scale. With MLOps, it is imperative to bring the different domains and personas together like the line of business (Business analysts), data (data engineers, data science); DevOps (ML Engineer, Application Developer/System programmers, Infrastructure engineers).

6.1.1 Optimizing MLOps with MLz

With IBM Z, we can build a modern scalable infrastructure, one that is tailor-made for transactional workloads. With MLz, we can deploy AI close to the transactional workloads and where the data originates and operationalize AI for accelerated business insights.

We can integrate a modern and robust MLOps process with MLz and overall IBM Z infrastructure. MLOps is a key requirement for our clients to operationalize AI on IBM Z. Figure 6-1 on page 115 illustrates an end-to-end MLOps pipeline with IBM Machine Learning on IBM Z.

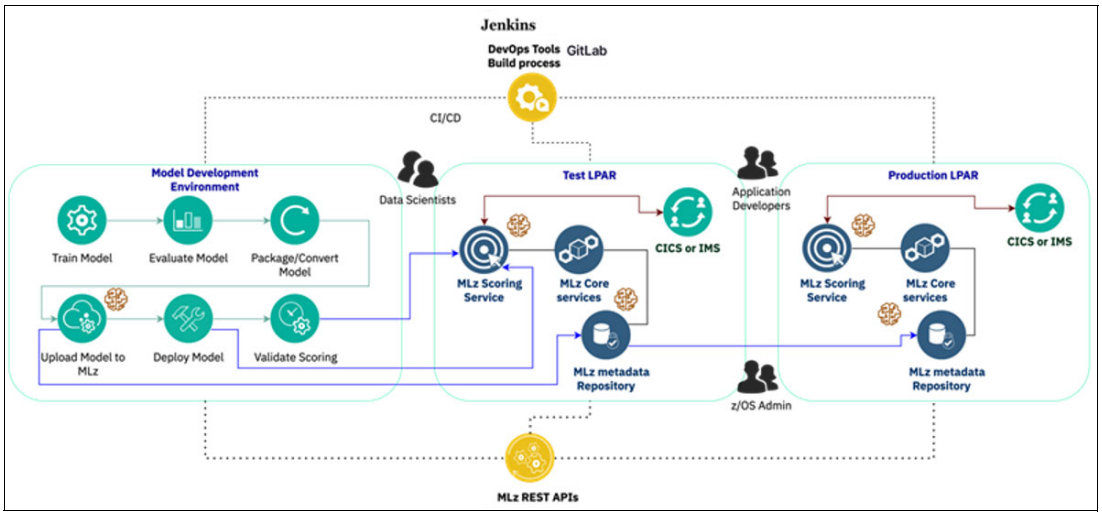


Figure 6-1 End-to-end MLOps pipeline for Machine Learning for IBM z/OS

MLz supports the MLOps process through restful interfaces. With the comprehensive set of APIs provided to support ML operationalization and the end-to-end process starting from data pre-processing, model training, model management and model deployment, to model scoring and continuous model monitoring.

In MLz v3.1, we further optimized the APIs with API Refine, user role and privilege control, along with updates to API documentation and examples. For clients, this means leveraging MLz APIs, they can easily integrate MLz into their existing CI/CD pipelines. You will learn the details of the MLz, CI/CD pipeline integration process in the next few sections.

Usually, the machine learning lifecycle architecture consists of the following three stages:

- Development stage
- Pre-production stage
- Production stage

The lifecycle stages include receiving code updates, training, deploying, and monitoring models.

Development stage

As illustrated in Figure 6-2 on page 116, an orchestrated development pipeline typically includes the following steps:

1. Data connection and validation
2. Data preparation
3. Model training and evaluation
4. Model deployment
5. Model validation (optional)

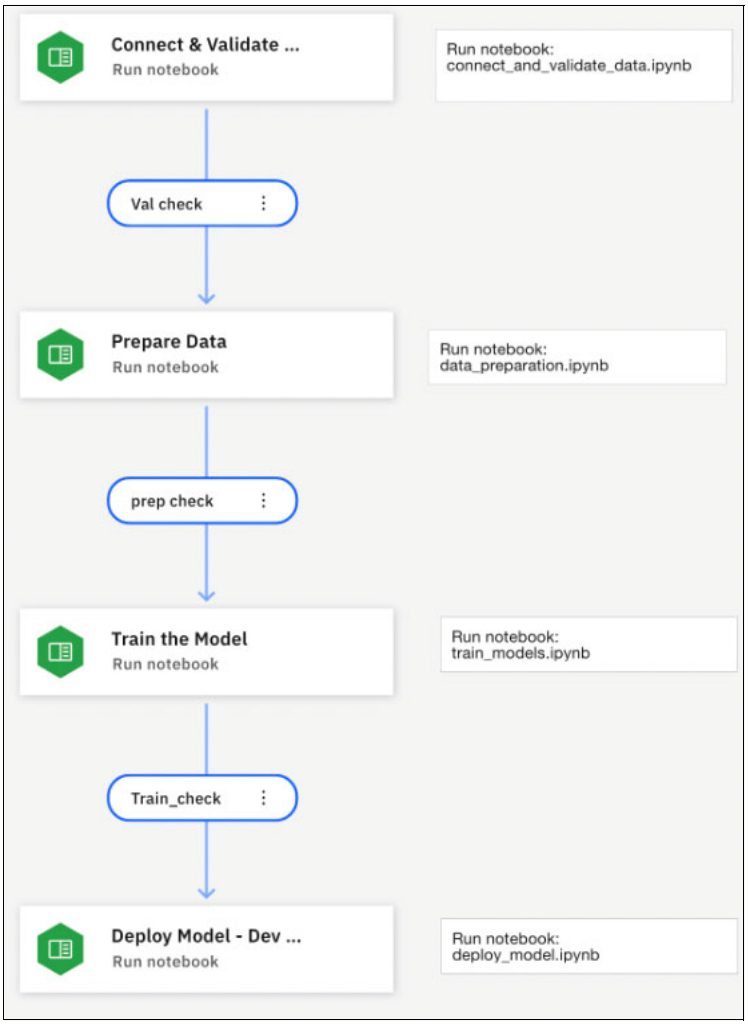


Figure 6-2 Sample development pipeline pulled from IBM Cloud Pak for Data

The output scripts tend to be Python scripts in Jupyter notebooks. They are version controlled using Git and serve as one of the components in the pre-production pipeline.

Pre-production stage

When there is a change committed to the Jupyter notebooks and a pull request is made, Jenkins starts the integration tests. As the integration tests pass, administrators merge the changes and Jenkins triggers the pre-production pipeline (see Figure 6-3 on page 117) that is generally comprised of the following steps:

1. Data extraction and data validation
2. Data preparation
3. Model training and model evaluation
4. Model deployment (pre-production space)
5. Model monitoring and model validation

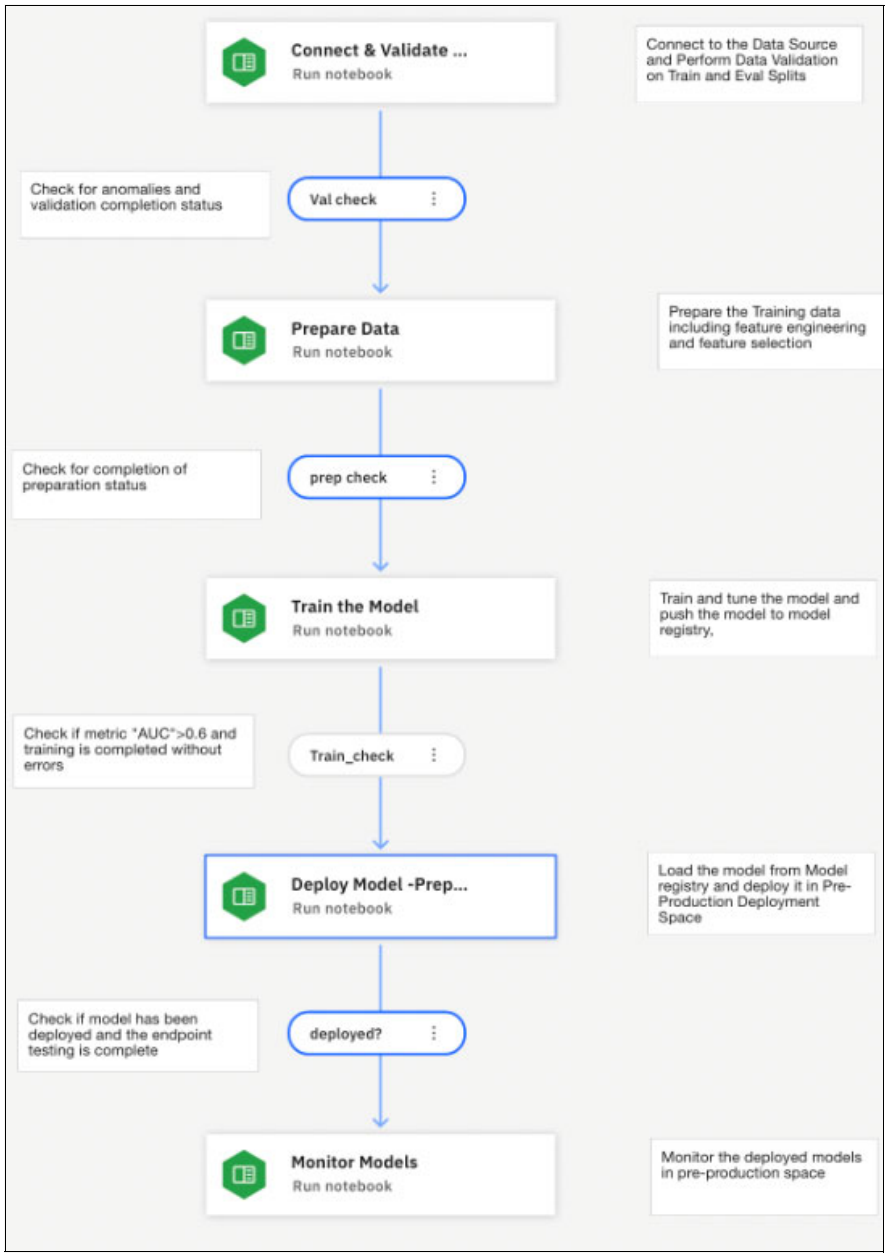


Figure 6-3 Sample pre-production pipeline pulled from IBM Cloud Pak for Data

Production stage

An orchestrated production pipeline, as illustrated in Figure 6-4 on page 118, typically contains following steps:

1. Deployment checks
2. Model deployment (production space)
3. Model monitoring
4. Model retraining

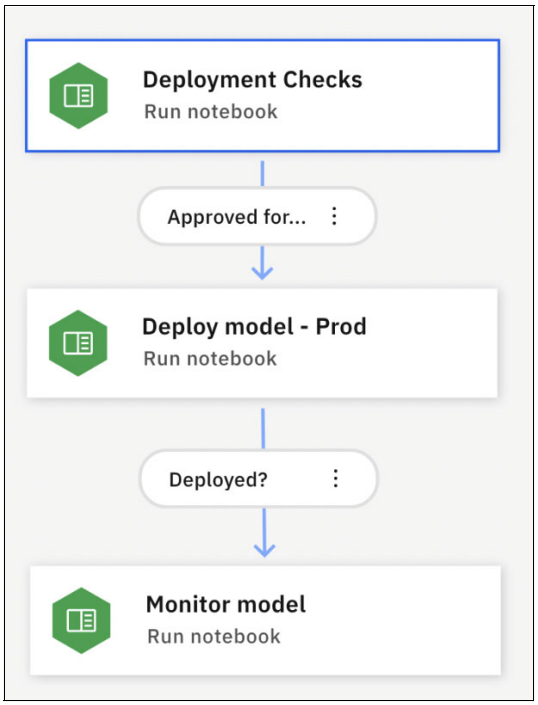


Figure 6-4 Sample production pipeline pulled from IBM Cloud Pak for Data

Notice that most of the steps throughout the three stages of development, pre-production, and production are the same, despite falling under three distinct pipelines. We will discuss these steps in greater detail in the upcoming sections.

6.2 Model development - model training and tuning

Model training and tuning constitute the foundational phase of the machine learning lifecycle, where data and algorithms synergize to create predictive models that can make informed decisions on new, unseen data. This phase involves a series of iterative steps that encompass data preprocessing, algorithm selection, training, validation, and fine-tuning. We will discuss the flow of this phase at a high level, as depicted in Figure 6-5 on page 119.

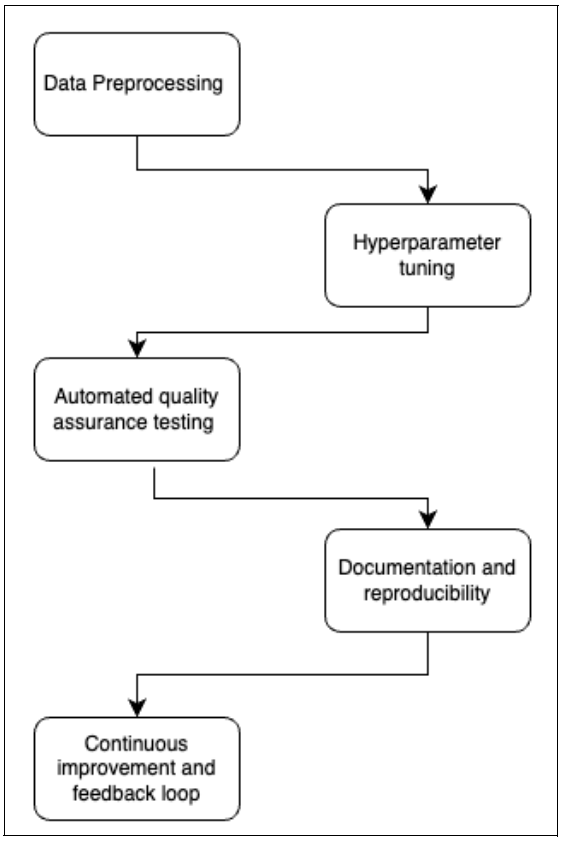


Figure 6-5 Flow for model training and tuning

6.2.1 Data preprocessing

Data preprocessing is the process of data mining where data is treated as raw products, such as raw crude oil. Various processes convert the crude oil to refined oil. Similarly, various processes need to be applied to raw data to convert it to a more refined data. These various processes include data cleansing, transforming, and data conversion, to name a few. Chapter 3, “The art of data engineering” on page 23 of this book discusses data preprocessing in detail, what those steps are, and how they come together.

6.2.2 Hyper parameter tuning

The model training process utilized multiple data sets for the model. Each data set and model need a different set of hyper parameters to find the correct parameter to be used by the model. Numerous iterations of model execution are done with a variety of parameters. Every parameter needs to run through the model end-to-end. This entire process is called hyper parameter tuning and runs simultaneously with the model. This hyper parameter tuning process is a combination of manual and automated process.

Hyper parameters are the combination of external parameters, an external variable for the model. An example of a hyper parameter is the number of nodes and layers in the neural network and decision branches.

Hyper parameter selection is a very important and essential process, critical for model performance. This is a trial-and-error process because each model is different, and only one set of rules can be applied at a time, so multiple attempts are required.

6.2.3 Automated quality assurance testing

Model training and tuning repetitive process. The model needs to be kept tuning, and retraining is in case source data changes. The model might be seeing data that the model has not seen before. A lot of new data could be a better thing for the model, and that means the model needs to retrain and tune it again and again. It is quite a repetitive process, but most steps can be automated; automation will produce reliable results without introducing human error.

The automated test process is part of MLOps, where data and model output are monitored and based on the model output. Automated retraining and tuning processes can be triggered based on the model's accuracy. Once the model is retrained and tuned, the automated testing process can be streamlined, and the entire model is validated with reality and without any human intervention.

6.2.4 Documentation and reproducibility

The trustworthy ML model has four major pillars: Explainability, Regulation, Integrity, and Reproducibility. Reproducibility is very important for the model trustworthy AL/ML model. Reproducibility is the process of producing ML model output by following the same workflow and giving the same output. Every execution run should produce a similar result every time. Reproducibility also has some challenges, such as changes in the data set, no proper logging, changes in hyper parameters, and changes in the model environment. The entire model workflow should be documented with evidence for explainability. The model output should explain how it has reached the given model output.

6.2.5 Continuous improvement and feedback loop

Once the model is deployed to the production environment, it must be monitored continuously. Various factors, such as changes in input data, runtime environment changes, and stale data, can impact model output. Model feedback should trigger the model retraining and tuning process.

The next section talks in detail about MLOps under the model review and deployment section. ML ops can monitor the model regularly with feedback methods. MLOps can monitor the model output and ensure it produces reliable output.

6.3 Model review and deployment

The step of model assessment and deployment emerges as the pivot that bridges the gap between model development and the tangible impact of MLOps as illustrated in Figure 6-6 on page 121. This critical stage marks the transition of complex algorithms from conceptual wonders to practical realities, ensuring that they not only perform admirably, but also seamlessly align with organizational goals.

The model review and deployment step are essentially a synchronization of examination and validation. It involves an in-depth review of the model's architecture, code integrity, and generalization abilities. This comprehensive review assures that the model's findings are not limited to the laboratory and can be applied in relevant, real-world applications.

This section outlines approaches for deploying accurate, ethical, and safe models that align with business needs during the MLOps journey. It emphasizes the importance of automation

for deployment consistency and investigates a range of deployment strategies, from microservices to containers, and the factors that influence their selection.

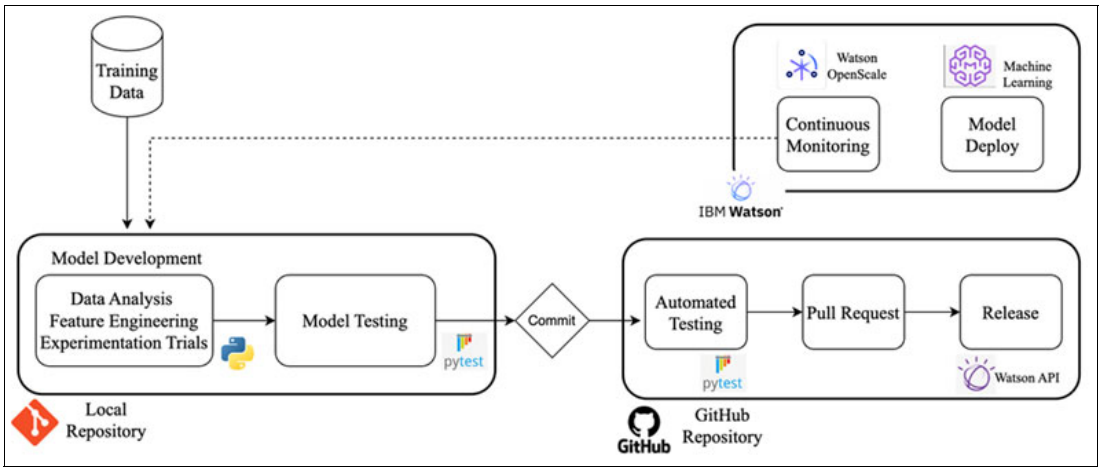


Figure 6-6 Model development lifecycle

6.3.1 Model review and validation

In the journey from model development to real-world application, the phase of model review and validation plays a crucial role in ensuring that the models not only work, but also work well. You can think of this phase as the thorough inspection before launching a rocket ship - it is focused on thoroughly checking every part of the aircraft to make sure it is ready for the journey ahead.

Bringing it back down to Earth to model development and applications, in order to adequately prepare a model from development to real-work application, the model review and validation phase iterates through the following processes:

- Code review and validation - Think of the model's code as the blueprint of a complex structure. During code review, teams collaborate to carefully examine the code for any mistakes or problems. It is like going through a checklist to ensure everything is in the right place. This step helps catch errors early and guarantees that the model's foundations are solid.
- Unit testing - Imagine testing individual puzzle pieces before assembling them into a complete picture. Unit testing involves running small tests on different parts of the model's code to ensure they work as expected. If any piece doesn't fit right, it is fixed before it becomes a part of the whole. This ensures that each component functions properly.
- Model validation - Validating a model is like sending it through a trial run. The model is tested on data it hasn't seen before to check if it can make accurate predictions. This process ensures that the model can handle new situations and is not just memorizing what it has seen before. Validating also helps identify if there's any bias or unfairness in the model's decisions, ensuring it treats everyone fairly.
- Addressing bias - Models can sometimes unintentionally make unfair decisions based on historical data. Addressing bias means actively working to make sure the model's predictions are fair for everyone. This is a vital step in ensuring that the model doesn't perpetuate or amplify existing inequalities.

In summary, the model review and validation phase act as the quality control checkpoint, helping guarantee that the model is robust, unbiased, and ready for deployment. A critical

step to ensuring the model's performance aligns with real-world expectations and ethical standards before it enters the operational stage.

6.3.2 Model versioning

Model versioning is a critical practice within the realm of MLOps that involves keeping track of different iterations, or versions, of a machine learning model. Just as software developers use version control systems to manage code changes, model versioning allows data scientists and MLOps engineers to systematically track and manage changes made to a model's architecture, code, and configuration over time.

The following terms and their descriptions are provided from the context of model versioning.

- Tracking progress

  Each model version represents a snapshot of the model's state at a specific point in time. This helps teams track the evolution of the model from initial development to refined iterations, capturing improvements, bug fixes, and enhancements along the way.

- Reproducibility

  Model versioning ensures that a specific model version can be accurately reproduced whenever needed. This is crucial for maintaining consistency in research, development, and production environments.

- Experimentation

  Data scientists often experiment with various algorithms, hyper parameters, and data pre-processing techniques. Model versioning allows them to keep a record of these experiments and the corresponding model performance.

- Collaboration

  In collaborative environments, model versioning enables team members to collaborate effectively. Team members can easily share, review, and work on different versions of the model, enhancing productivity and knowledge sharing.

- Comparative analysis

  With multiple model versions available, teams can perform comparative analysis to understand how changes impact performance. This aids in making informed decisions about which version to deploy.

- Deployment and rollback

  When deploying models into production, having well-managed versions simplifies the deployment process. If a new version presents unexpected issues, rolling back to a previous version is straightforward.

You can use the model management APIs to manage the models that are stored in MLz. You can use the following API to get the metadata of all the versions of a model,

```
GET /v3/ml_assets/models/{model_id}/versions
```

For more information on model management APIs, see the following IBM documentation:

https://www.ibm.com/docs/en/wml-for-zos/enterprise/3.1.0?topic=specification-model-management-apis

To implement effective model versioning, teams can use version control tools, such as Git, combined with specific practices tailored to machine learning. This might involve tagging model versions, documenting changes in a version control system, and integrating versioning within the MLOps pipeline.

6.3.3 Integrating with CI/CD pipelines

Much like DevOps, CI/CD (Continuous integration, continuous delivery) is a set of practices and steps (see Figure 6-7) that software and application development teams use to deliver code changes with speed and reliability leveraging automation.

In machine learning, these stages are different than in software development. Firstly, a model depends not only on the code, but also on the data and the hyper parameters. Additionally, deploying a model to production is more complex than deploying software to production. For a rapid and reliable update of pipelines in production, you need a robust automated CI/CD system. With automated CI/CD pipelines, data scientists can rapidly explore new ideas around feature engineering, model architecture, and hyper parameters.

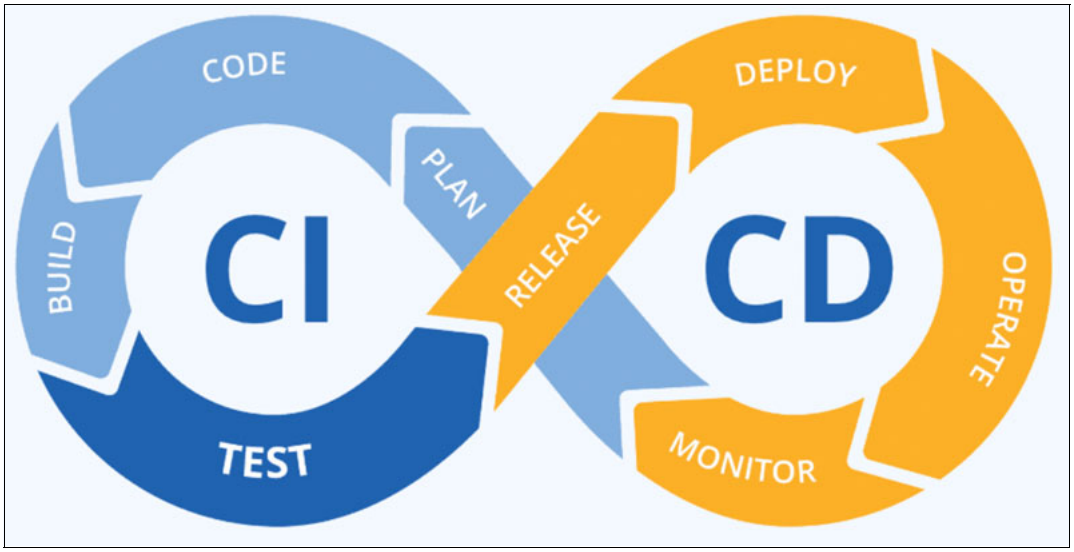


Figure 6-7 Stages of a CI/CD pipeline

Continuous Integration (CI)

Continuous integration is a software development process where developers integrate the new code, they've written more frequently throughout the development cycle. Automated testing is done against each iteration of the build to identify integration issues earlier, when they are easier to fix, which also helps avoid problems at the final merge for the release. Overall, continuous integration helps streamline the build process, resulting in higher-quality software and more predictable delivery schedules.

Continuous integration within the realm of machine learning implies that whenever code or data undergo updates, the machine learning pipeline is re-invoked. This process is orchestrated in a manner where versioning and reproducibility are upheld, allowing seamless codebase sharing across various projects and teams. Each pipeline rerun may encompass activities like training, testing, or generating updated reports, thereby simplifying the task of comparing against other versions in production.

The following instances illustrate the CI workflow:

- Executing and versioning training and assessment for each repository commit.
- Executing and contrasting experimental runs for every Pull Request towards a specific branch.
- Initiating new runs at regular intervals.

Continuous Deployment (CD)

Continuous deployment stands as an automated approach for rolling out new releases into production or other environments like staging. This approach facilitates swift and ongoing changes, enabling prompt user feedback and accommodating new data for model retraining or novel models.

The following instances exemplify the CD workflow:

- Ensuring the infrastructure environment meets requisites prior to deployment.
- Validating model outcomes using established inputs.
- Conducting load tests and evaluating model latency.

CI/CD tools and configuration

When selecting CI/CD tools, the focus should be on how to optimize and automate the software development process. An effective CI/CD pipeline uses open-source tools for integration, testing and deployment. Correct configuration of your CI/CD process also impacts the success of the software development pipeline.

The most common open-source CI/CD tool is Jenkins. Jenkins is an automated CI server written in Java and used for automating CI/CD steps and reporting. Other open-source tools for integration include Travis CI and CircleCI.

Integrated development environments (IDE), such as GitHub or AWS CodeCommit, help developers create, maintain and track software packages, while platforms like GitLab seek to provide the IDE within a comprehensive platform that includes other tools.

When operating in a cloud environment, teams use containers like Docker for packaging and shipping applications, and they use Kubernetes for orchestration. While Kubernetes is not strictly for the CI/CD pipeline, it is used in many CI/CD workflows.

CI/CD pipeline phases

From source code to production, the following phases make up the development lifecycle and workflow of the CI/CD pipeline:

- Build - This phase is part of the continuous integration process and involves the creation and compiling of code. Teams build off source code collaboratively and integrate new code while quickly determining any issues or conflicts.
- Test - At this stage, teams test the code. Automated tests happen in both continuous delivery and deployment. These tests could include integration tests, unit tests, and regression tests.
- Deliver - Here, an approved codebase is sent to a production environment. This stage is automated in continuous deployment and is only automated in continuous delivery after developer approval.
- Deploy - Lastly, the changes are deployed, and the final product moves into production. In continuous delivery, products or code are sent to repositories and then moved into production or deployment by human approval. In continuous deployment, this step is automated.

6.3.4 Model deployment

Deploying a machine learning model into a production environment requires careful consideration of various factors, including scalability, resource utilization, response time, and ease of maintenance. Different deployment strategies offer unique approaches to making

your model accessible and operational. Let us explore some common deployment strategies – batch processing and online deployment.

Batch processing

In some cases, it is more efficient to perform model inference on a batch of data rather than real-time processing. Batch processing involves scheduling model inference on a batch of data at specific intervals. This strategy is suitable for scenarios where low latency is not a priority, such as generating reports or analyzing historical data. Examples include insurance premium and claims analytics, sales and inventory analysis, log analysis and anomaly detection, and credit scoring and loan approval.

An example of batch processing is discussed in 5.3.2, “Interfaces of MLz online scoring service” on page 84.

Online Deployment

An online deployment strategy, also known as real-time deployment, involves making machine learning models available for inference and predictions in real-time as new data arrives. This strategy is particularly useful for applications where quick and immediate responses are required based on incoming data. Examples include fraud-detection, chatbots, recommendation systems, image recognition in video streams, and IoT applications.

You can use following API to create online deployment for a model:

```
POST /v3/published_models/${modelId}/deployments
```

You can also use following API to get the detailed information of a deployment:

```
GET /v3/published_models/${modelId}/deployments/${deploymentId}
```

For more information on model deployment APIs, see the following IBM documentation:

https://www.ibm.com/docs/en/wml-for-zos/enterprise/3.1.0?topic=specification-deployment-apis

When choosing a deployment strategy, consider factors such as the nature of your application, expected traffic patterns, resource requirements, and deployment complexity. Each strategy has its advantages and challenges, so selecting the one that aligns best with your project goals and constraints is crucial. Additionally, the availability of tools and platforms that support your chosen strategy can simplify the deployment process and enhance its scalability and reliability.

6.3.5 Model inference and scoring

Model inferencing and scoring represent the bridge between machine learning models and tangible, real-world impact. In the dynamic landscape of MLOps, this phase demands meticulous attention to responsiveness, interpretability, scalability, and security. By seamlessly integrating these considerations, organizations can harness the full potential of their models, making AI-driven decisions that are not only accurate but also ethical, secure, and aligned with business objectives.

Once you have created an online deployment for a model, we can access its predictions by sending requests to its scoring endpoint.

You can find the detailed procedure for making online predictions using scoring APIs with MLz at the following IBM Documentation:

https://www.ibm.com/docs/en/wml-for-zos/enterprise/3.1.0?topic=specification-scoring-apis

To automate the whole inferencing process, you can use the sample Python functions, provided in Example 6-1, to interact with MLz through REST APIs.

Example 6-1 Using Python functions to automate the inferencing process

```
# Get MLz user token
def gen_wmlz_user_token(user_name, password):
    gentoken_url = '{}:{}/auth/generateToken'.format(
            wmlz_service_host, wmlz_service_port)
    gentoken_request_header = {
         'Content-Type': 'application/json'
    }
    gentoken_request_data = json.dumps({'username': user_name, 'password':
password})
    gentoken_response = session.post(gentoken_url, gentoken_request_data,
headers=gentoken_request_header, verify=False)
    gentoken_response_dict = json.loads(gentoken_response.content)
    return gentoken_response_dict['token']

# Retrieve model ID and model version ID of the uploaded model
def get_model_info(model_name):

    get_model_info_url = '{}:{}/v3/ml_assets/models?modelName={}'.format(
            wmlz_service_host, wmlz_service_port, model_name)
    get_request_header = {
            'Authorization': service_token
        }
    get_model_info_response = session.get(get_model_info_url,
headers=get_request_header, verify=False)
    print("get model info result:" + str(get_model_info_response.content))
    get_model_info_response_dict = json.loads(get_model_info_response.content)
    model_id =
get_model_info_response_dict.get('resources')[0].get('metadata').get('guid')
    model_version_id =
get_model_info_response_dict.get('resources')[0].get('entity').get('model_version'
).get('guid')
    return model_id, model_version_id

# Perform online scoring
def do_score(scoring_url, scoring_input):

    score_request_header = {"Authorization": service_token, "Content-Type":
"application/json"}
    score_response = session.post(scoring_url, headers=score_request_header,
data=scoring_input, verify=False)
    score_response_dict = json.loads(score_response.content)
    return score_response_dict
```

6.4 Model monitoring

Model monitoring is a crucial phase within the machine learning operations (MLOps) framework that focuses on the ongoing observation, analysis, and management of deployed models in real-world environments. As models transition from controlled development settings

to dynamic production ecosystems, monitoring becomes the sentinel that guards against performance degradation, concept drift, anomalies, and other operational challenges.

- Model performance - The capability to assess a model's performance through a set of metrics and recording its decisions or outcomes can offer valuable directional insights. These insights can be contrasted with historical data, aiding in comparisons among different models to determine the most effective one.
- Data and security - Modern models rely on intricate feature pipelines and automated workflows, which involve dynamic data subjected to diverse transformations. Given the complexity, data inconsistencies and errors can unintentionally erode model performance over time. Additionally, models can be vulnerable to attacks through data injection and other means.
- Explainability - The opaque nature of models poses challenges in understanding and debugging, particularly within a production setting. The ability to elucidate a model's decision is crucial not only for enhancement but also for accountability, especially in sectors like finance.
- Addressing bias - As machine learning models glean relationships from training data, they may inadvertently magnify existing biases or introduce new ones. Detecting and mitigating bias during the development phase is intricate yet imperative.
- Monitoring for drift - The statistical characteristics of the target variable, which the model seeks to predict, can evolve unexpectedly over time. This phenomenon, known as concept drift, leads to decreasing prediction accuracy as time progresses, posing challenges that necessitate monitoring and mitigation.

6.4.1 Evaluating and reevaluating a model under MLz

As your data changes over time, the quality of the predictions from a particular model might degrade. With the WML for z/OS, you can evaluate a SparkML or PMML model and schedule it for periodic re-evaluations to ensure its accuracy. Evaluating models using MLz consists of following steps:

1. Create evaluation
2. Deploy models
3. Schedule evaluation

Figure 6-8 on page 128 is an example of a successful evaluation, where the Area under curve (AUC) was more than 0.75. Since, AUC is well above random chance, our model is considered to be accurate.

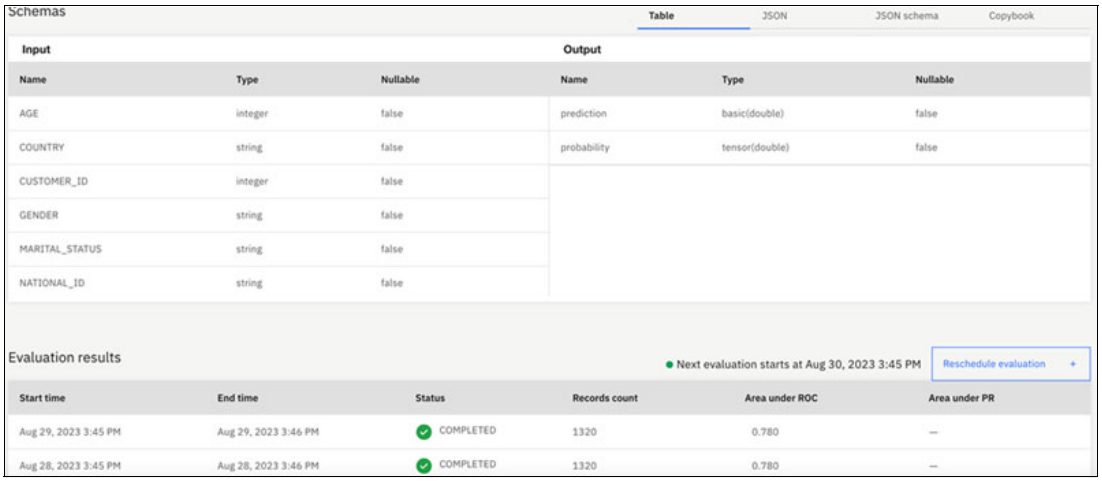


Figure 6-8 Evaluating a model using MLz

For more information on evaluating and reevaluating a model using MLz, see the following IBM Documentation:

https://www.ibm.com/docs/en/wml-for-zos/enterprise/3.1.0?topic=wmlz-evaluating-reevaluating-models

6.4.2 Monitoring deployed models

IBM Watson OpenScale offers a high-caliber setting designed for AI applications. It is possible to set up MLz to collaborate seamlessly with the Watson OpenScale service within the IBM Cloud Pak for Data ecosystem. You can use it to understand how your AI models make decisions, detect, and mitigate bias and drift, increase the quality and accuracy of your predictions, explain transactions, and perform what-if analysis. The capabilities of Watson OpenScale can be harnessed to validate MLz models employed in real-time scoring via the REST API.

Using IBM Watson OpenScale to monitor your AI models deployed in z/OS environment with trust and transparency

When the model is ready, then the data scientist pushes the model to the production environment, which is also known as business line environment. Then, the data scientist can use Watson OpenScale to check whether the z/OS AI model is experiencing model drift or not; and the people in the business, such as a loan officer or a loan customer, can use Watson OpenScale to understand the reason for the model results.

Step 1. Get z/OS machine learning model scoring endpoint from MLz

1. Log in to Machine Learning for IBM z/OS.
2. Click the **Deployments** tab.

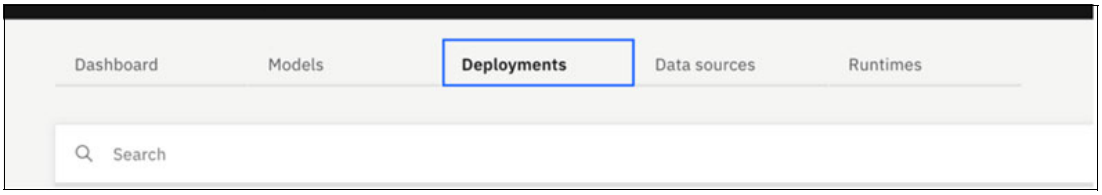


Figure 6-9 MLz model Deployments list

3. Get the scoring endpoint.

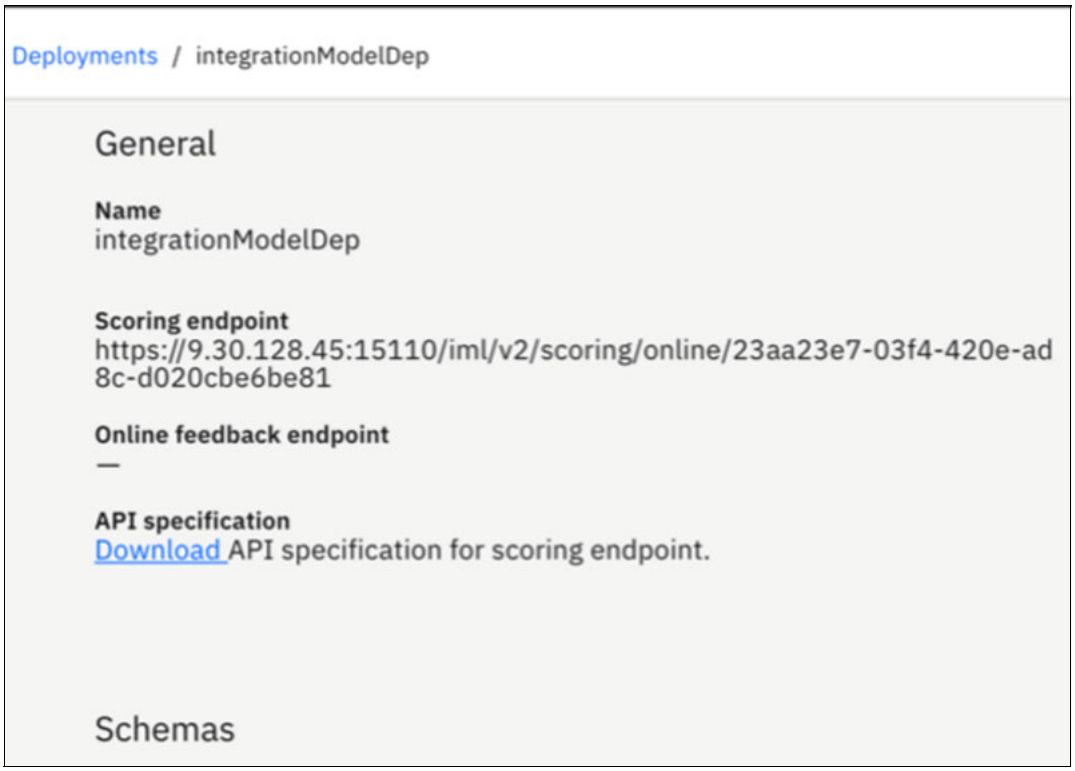


Figure 6-10 Deployed model in MLz

Step 2. Create a production machine learning provider in Watson OpenScale on the IBM Cloud Pak for Data platform

1. In the Watson OpenScale System setup page, click **Add machine learning provider** and then bind Machine Learning for IBM z/OS as the machine learning provider, populating the Connection details tag with the following information:
   - Service provider: **Custom Environment**
   - Authentication type: **Basic**
   - Username: `MLz username`
   - Password: `MLz user password`
   - Environment type: **Production**
2. Then click the **Save** button.

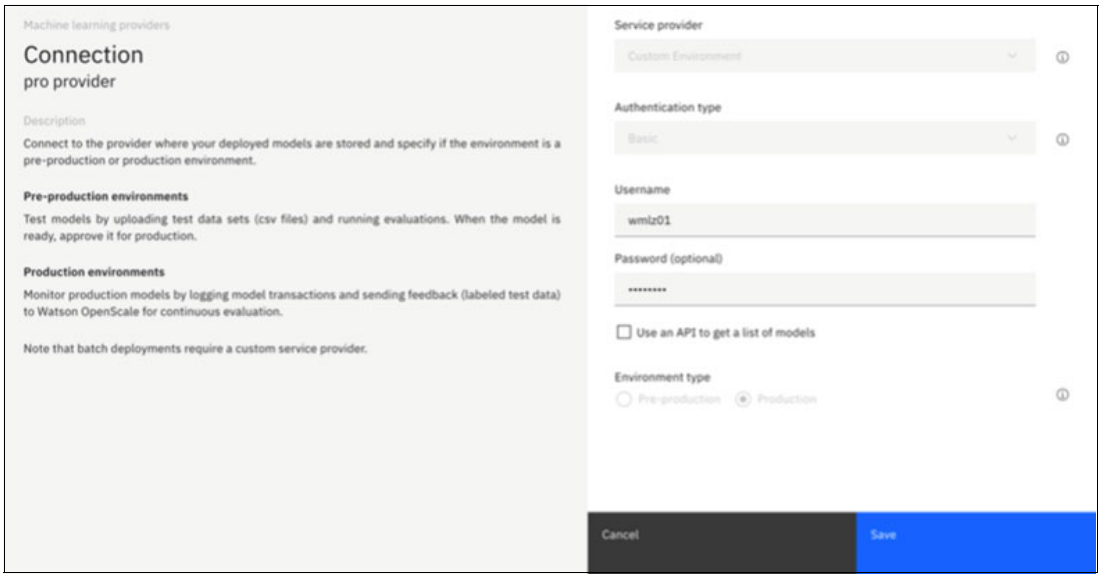


Figure 6-11 Connection details in Watson OpenScale

3. On the Watson OpenScale Insight dashboard, create a subscription base for the machine learning service provider you just created, then bind the Watson OpenScale subscription to the Machine Learning for IBM z/OS deployment, as follows:
   a. From the drop-down menu, select the correct Machine learning Provider and environment from step 2.1.
   b. In the Endpoint field, add the Machine Learning for IBM z/OS scoring endpoint from step 1.3.

   Note: The endpoint here should use the MLz UI port number. For example, if the scoring endpoint in the deployment is, `https://*.*.*.*:15110/iml/v2/scoring/online/23aa23e7-03f4-420e-ad8c-d020cbe6be81` and base URL is `https://*.*.*.*:15101/`, then the endpoint for the Watson OpenScale provider would be, `https://*.*.*.*:15101/iml/v2/scoring/online/23aa23e7-03f4-420e-ad8c-d020cbe6be81`.

   c. Fill in the Deployment Name.
   d. Click **Configure**.

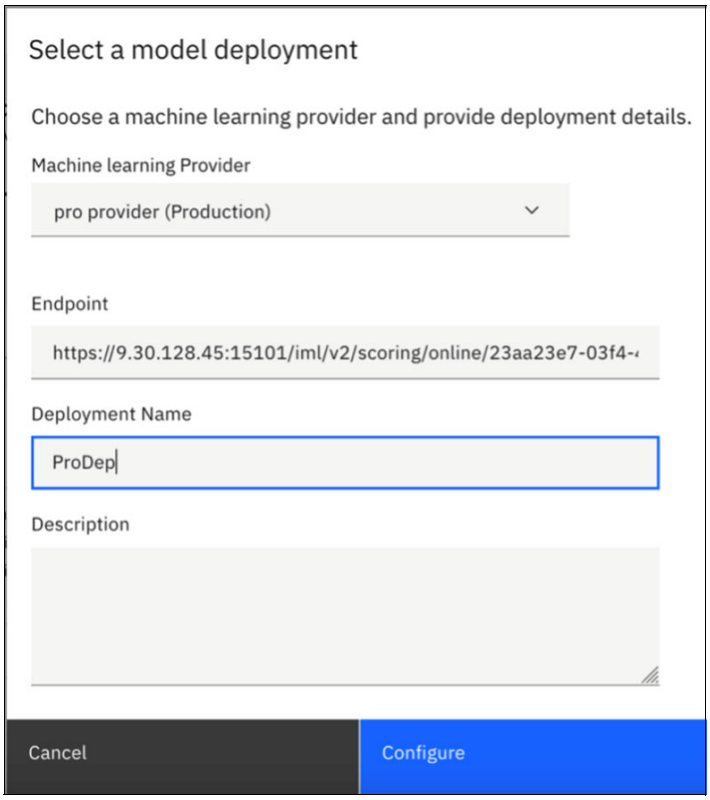


Figure 6-12 Watson OpenScale model deployment details

4. Configure the Watson OpenScale monitors - Fairness, Quality, Drift, and Explain.
5. Get the OpenScale Datamart ID and Subscription ID from the Endpoints page.

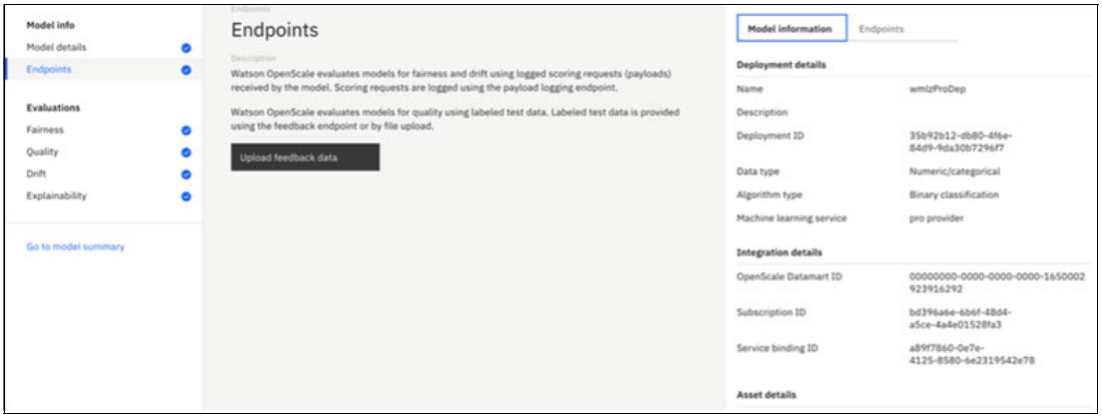


Figure 6-13 Model information details in Watson OpenScale

Enable and configure Watson OpenScale on Machine Learning for IBM z/OS

1. Enable Watson OpenScale in the Machine Learning for IBM z/OS deployment:
   a. Right-click on the **Action icon** under IBM Machine for z/OS deployment.
   b. Select **Configure for OpenScale**.

2. Configure Watson OpenScale with the following details:
   a. Select **Enable**
   b. Instance: `CP4D UI URL`
   c. User: `OpenScale user`
   d. Password: `OpenScale user password`
   e. DataMartID: `OpenScale Datamart ID` (from Step 2)
   f. Target ID: `Subscription ID` (from Step 2)
3. Then click **OK.**

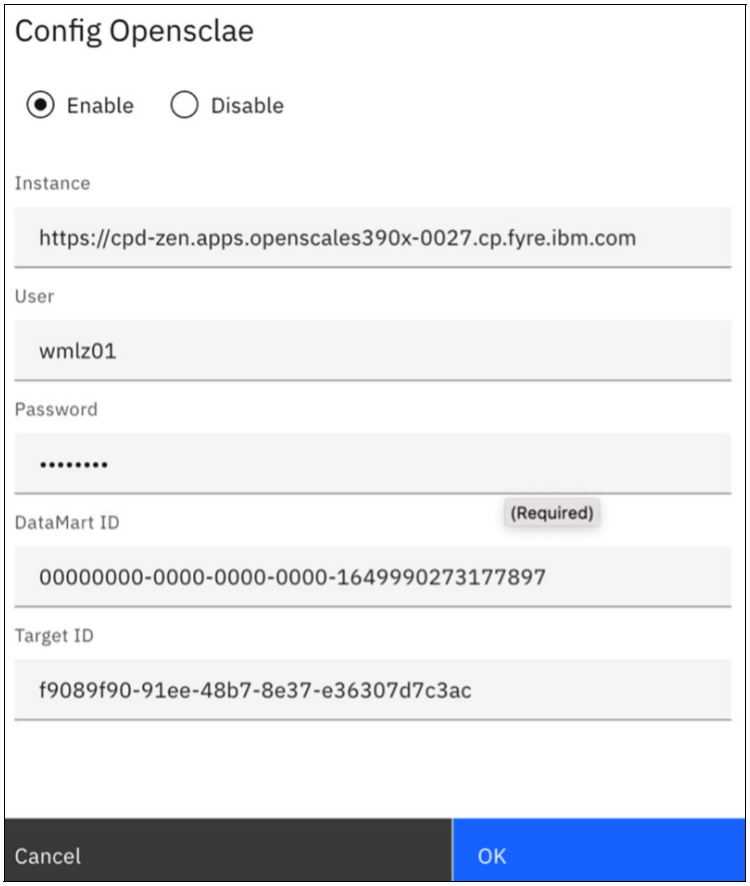


Figure 6-14 Configuration settings in Watson OpenScale

4. Evaluate the Machine Learning for IBM z/OS model for Fairness, Quality, Drift, and Explain with Watson OpenScale.

   Once Watson OpenScale has been configured and enabled, you can load the Insights Dashboard, which contains tiles for any models being monitored (see Figure 6-15 on page 133).

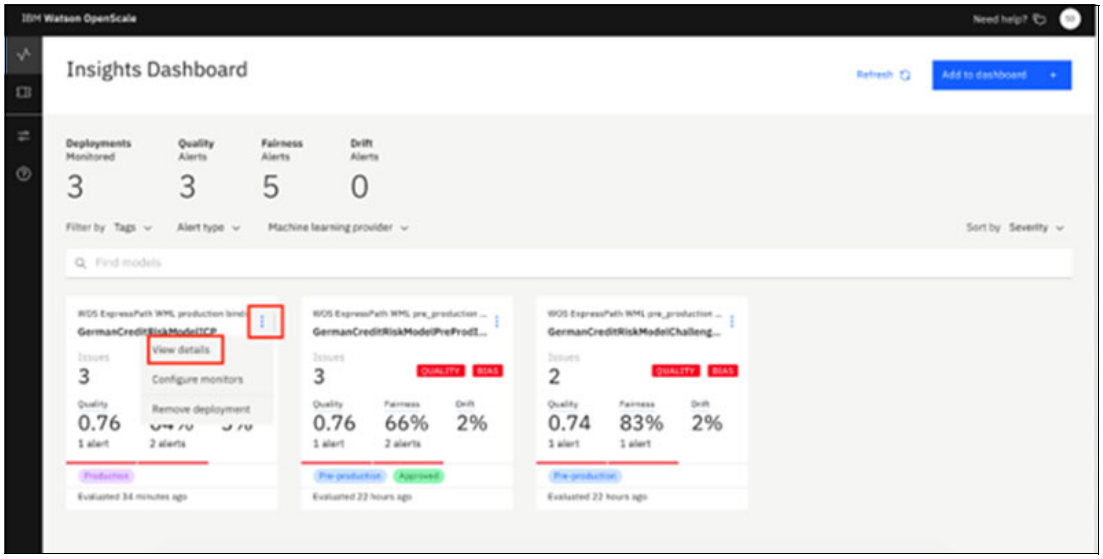


Figure 6-15 Watson OpenScale Insights Dashboard

You can then evaluate the models on the various set monitors, such as fairness, quality, and drift (see Figure 6-16).

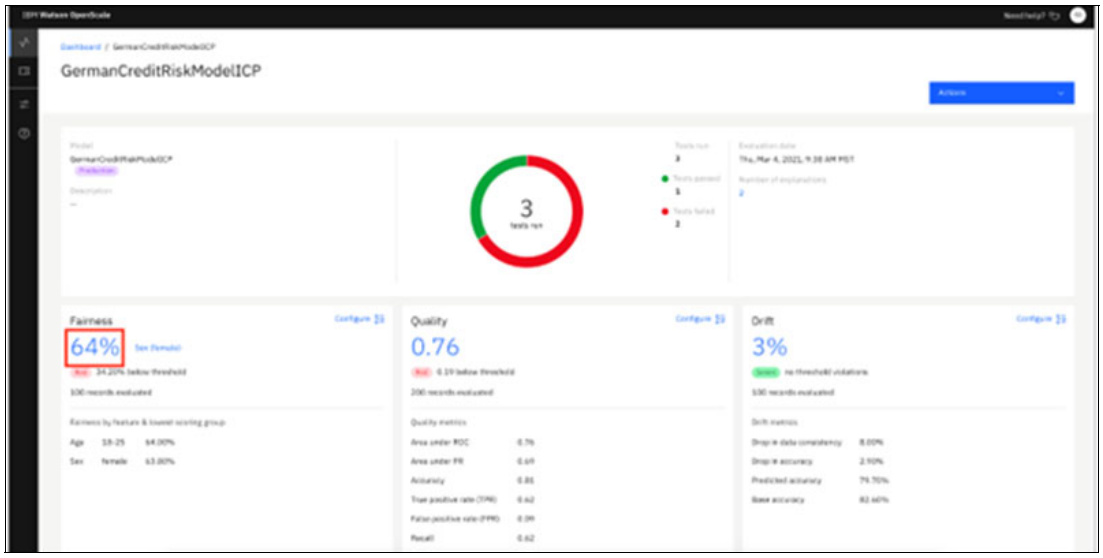


Figure 6-16 Evaluation monitors with Watson OpenScale

To understand your model's evaluation results better, you can dive deeper into each evaluation monitor (see Figure 6-17 on page 134).

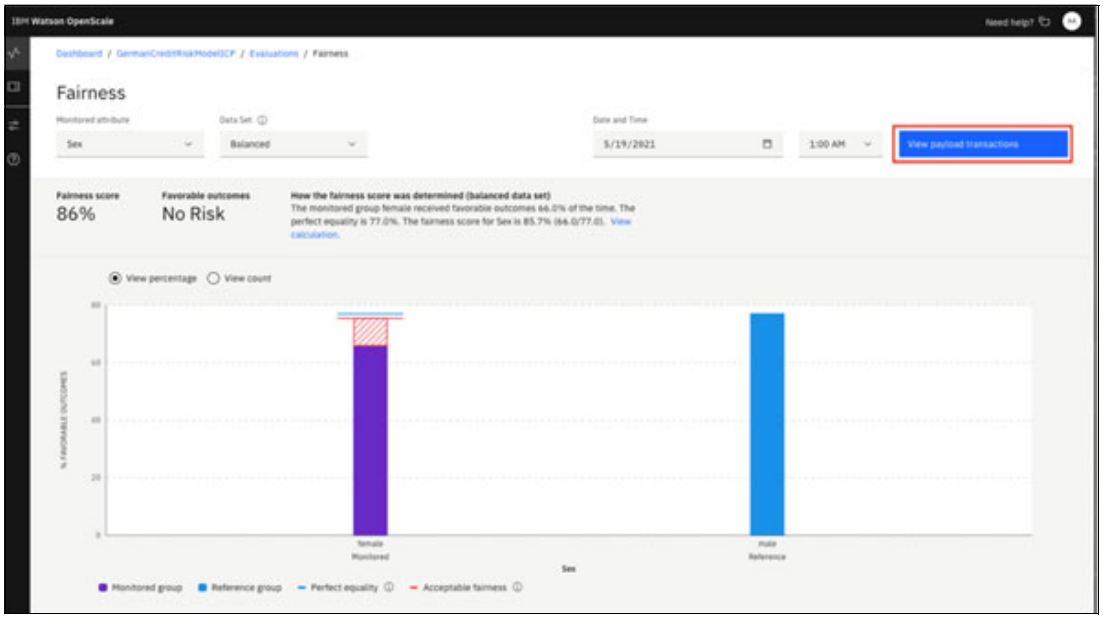


Figure 6-17 Watson OpenScale Evaluation monitor - Fairness

You can also explain the predictions of the model, where in you can see the relative weights of the most important features for the prediction (see Figure 6-18).

Figure 6-18 Watson OpenScale Explainability feature

Model monitoring requires the integration of various tools, technologies, and processes into the deployment pipeline. Automated monitoring systems, combined with human oversight, ensure that models continue to deliver accurate, fair, and reliable predictions even as real-world conditions evolve.

In summary, model monitoring is the vigilant guardian that ensures deployed models remain effective and aligned with operational requirements. It bridges the gap between model development and ongoing production, transforming models from static artifacts into dynamic, adaptable, and valuable assets within the MLOps ecosystem.

6.5 AI governance and security

AI governance and security are paramount considerations when deploying machine learning models into production environments. As organizations increasingly rely on AI for critical decision-making, it becomes imperative to ensure ethical, legal, and secure practices are in place throughout the model's lifecycle. AI governance and security are important in a MLOps lifecycle for the following reasons:

- Ethical AI Governance: Ethical AI governance stands as a sentinel guarding against the unintentional perpetuation of biases or discriminatory behaviors within AI models. It involves:
  - Bias Detection and Mitigation: Mechanisms to detect and rectify biases in training data, ensuring that models make fair and equitable predictions.
  - Fairness Assessments: Continuous assessments to gauge model fairness, preventing decisions that favor one group at the expense of others.
  - Transparency and Explainability: Employing techniques like model agnostic algorithms to provide transparent and interpretable explanations for model predictions, fostering trust and accountability.
  - Regulatory Compliance: The regulatory landscape surrounding AI is evolving rapidly. AI governance practices involve:
  - Data Privacy Compliance: Implementing stringent data protection measures to comply with regulations like GDPR or HIPAA.
  - Model Behavior Compliance: Ensuring that deployed models adhere to legal requirements, such as those governing financial decisions or health care treatments.
- Data Security: Data security is a linchpin in AI governance and security within MLOps:
  - Encryption and Access Controls: Employing robust encryption techniques and access controls to safeguard sensitive data used in model training and inferencing.
  - Secure Data Storage: Implementing secure storage protocols to protect data from unauthorized access or breaches.
- Incident Response and Accountability: Preparing for AI-related incidents is integral:
  - Incident Response Plans: Developing clear plans and protocols to address model failures, security breaches, or ethical lapses.
  - Audit Trails: Maintaining detailed audit trails that record model development, validation, and deployment processes for accountability and compliance purposes.
  - Access Control and Authentication: Strict access controls and authentication mechanisms ensure that only authorized personnel can interact with deployed models, preventing unauthorized usage or tampering.
  - Monitoring and Anomaly Detection: Automating continuous monitoring of deployed models to detect anomalies, security breaches, or performance degradation, enabling proactive responses.
  - Documentation and Compliance Audits: Comprehensive documentation of AI governance and security practices aids in compliance audits and regulatory checks, ensuring that all processes are aligned with legal requirements.

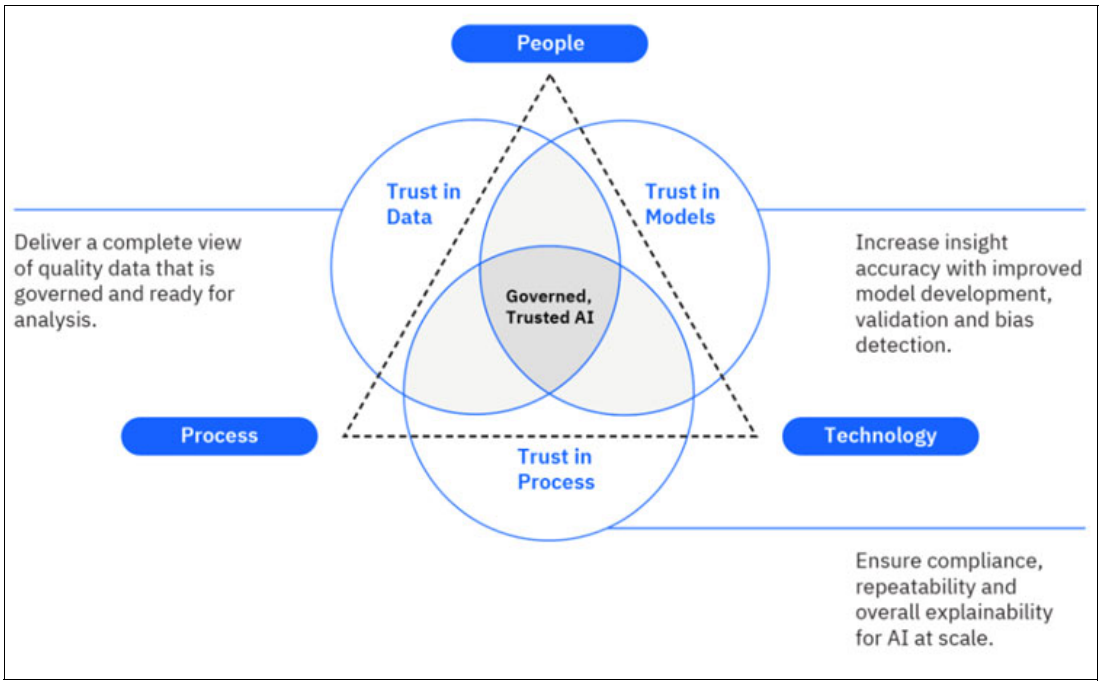


Figure 6-19 AI Governance on IBM Z for Trustworthy AI

AI governance and security are not just static considerations; they are dynamic and integral components of the MLOps journey. By integrating ethical practices, robust security measures, and regulatory compliance into every stage of MLOps, organizations can harness the power of AI while maintaining trust, transparency, and the highest standards of security and ethics.

6.6 How generative AI is evolving MLOps

The release of OpenAI's ChatGPT and IBM's WatsonX sparked interests in AI capabilities across industries and disciplines. This technology, known as generative AI, has the capability to write software code, create images and produce a variety of data types, as well as further develop the MLOps process.

Generative AI is a type of deep-learning model that takes raw data, processes it and "learns" to generate probable outputs. In other words, the AI model uses a simplified representation of the training data to create a new work that is similar, but not identical, to the original data. For example, by analyzing the language used by Shakespeare, a user can prompt a generative AI model to create a Shakespeare-like sonnet on a given topic to create an entirely new work.

Generative AI relies on foundation models to create a scalable process. As AI has evolved, data scientists have acknowledged that building AI models takes a lot of data, energy, and time, from compiling, labeling, and processing data sets the models use to "learn" to the energy is takes to process the data and iteratively train the models. Foundation models aim to solve this problem. A foundation model takes a massive quantity of data and using self-supervised learning and transfer learning can take that data to create models for a wide range of tasks.

This advancement in AI means that data sets are not task specific—the model can apply information it is learned about one situation to another. Engineers are now using foundation models to create the training models for MLOps processes faster. They simply take the

foundation model and fine-tune it using their own data, versus taking their data and building a model from scratch.

As you embark on your own MLOps journey, remember that it is not just about the models; it is about the impact those models have on the world. It is about streamlining the path from data to decisions, from algorithms to applications, and from insights to impact. Welcome to the world of MLOps, where the future is defined by the intelligent application of data-driven insights.

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

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *Solving challenges of instant payments by using AI on IBM zSystems*, REDP-5698
- *Accelerate Mainframe Application Modernization with Hybrid Cloud*, REDP-5705
- *Securely Leverage Open-Source Software with Python AI Toolkit for IBM z/OS*, REDP-5709

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- IBM z/OS Connect EE Product Page

  https://www.ibm.com/products/zos-connect-enterprise-edition
- IBM Data Virtualization Manager for z/OS Product Page

  https://www.ibm.com/products/data-virtualization-manager-for-zos
- IBM Cloud Pak for Data Product Page

  https://www.ibm.com/products/cloud-pak-for-data
- IBM Watson Knowledge Catalog

  https://www.ibm.com/cloud/watson-knowledge-catalog
- IBM DataStage Product Page

  https://www.ibm.com/products/datastage
- IBM z/OS v3.1 documentation - What is z/OS Container Extensions?

  https://www.ibm.com/docs/en/zos/3.1.0?topic=extensions-what-is-zos-container
- CELENT: Operationalizing Fraud Prevention on IBM z16 - Reducing Losses in Banking, Cards, and Payments

  https://www.ibm.com/downloads/cas/DOXY3Q94
- IBM Institute for Business Value's guide Generating ROI with AI - Six capabilities that drive world-class results:

  https://www.ibm.com/downloads/cas/DDORNOB2

- What is data refinery?

  https://www.ibm.com/products/data-refinery

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

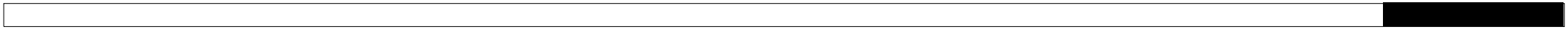
To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

(1.5" spine)
1.5"<-> 1.998"
789 <->1051 pages

(1.0" spine)
0.875"<->1.498"
460 <-> 788 pages

(0.5" spine)
0.475"<->0.873"
250 <-> 459 pages

(0.2"spine)
0.17"<->0.473"
90<->249 pages



(0.1"spine)
0.1"<->0.169"
53<->89 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

SG24-8552-00

ISBN DocISBN

Printed in U.S.A.