

Optimized Inferencing and Integration with AI on IBM zSystems

Introduction, Methodology, and Use Cases

Makenzie Manna

Erhan Mengusoglu

Artem Minin

Krishna Teja Rekapalli

Thomas Rüter

Pia Velazco

Markus Wolff



 Analytics

IBM Z



IBM Redbooks

**Optimized Inferencing and Integration with AI on IBM
zSystems: Introduction, Methodology, and Use Cases**

November 2022

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

Second Edition (November 2022)

This edition applies to z/OS Version 2, Release 5.

© Copyright International Business Machines Corporation 2022. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

| | |
|---|------|
| Notices | v |
| Trademarks | vi |
| Preface | vii |
| Authors | viii |
| Now you can become a published author, too! | ix |
| Comments welcome | x |
| Stay connected to IBM Redbooks | x |
| Summary of changes | xi |
| November 2022, Second Edition | xi |
| Chapter 1. Foundations of artificial intelligence | 1 |
| 1.1 What is artificial intelligence | 2 |
| 1.1.1 Artificial intelligence, machine learning, and deep learning | 3 |
| 1.2 Advantages and value of data and AI on IBM zSystems | 4 |
| 1.2.1 Delivering innovation with AI and IBM z16 | 4 |
| 1.2.2 Open-source frameworks on IBM zSystems | 7 |
| 1.3 Building and training anywhere and deploying and infusing on IBM zSystems | 8 |
| 1.3.1 Modernizing z/OS applications by infusing AI | 12 |
| Chapter 2. Methodology and tools | 17 |
| 2.1 Developing a machine learning model | 18 |
| 2.1.1 Defining the business problem | 18 |
| 2.1.2 Approaches and selecting an approach | 19 |
| 2.1.3 Machine learning lifecycle | 20 |
| 2.2 Integration and deployment best practices | 24 |
| 2.2.1 Model interoperability | 24 |
| 2.2.2 Piloting your first project | 25 |
| 2.2.3 IBM Watson Machine Learning for z/OS overview and integration points | 26 |
| 2.2.4 Batch versus online inference | 31 |
| 2.2.5 Continuous integration and continuous delivery in ML | 33 |
| Chapter 3. Real-time, in-transaction scoring use case scenarios | 37 |
| 3.1 Business value of in-transaction scoring | 38 |
| 3.1.1 Industry-specific scenarios that use in-transaction scoring | 38 |
| 3.1.2 Optimizing in-transaction decisions at scale with AI on IBM zSystems | 40 |
| 3.1.3 Solution architecture overview | 41 |
| 3.2 Credit risk assessment use case scenario | 48 |
| 3.2.1 Data science and credit risk model development | 48 |
| 3.2.2 Credit risk model deployment and evaluation on IBM zSystems | 55 |
| 3.2.3 CICS application integration for real-time credit risk scoring | 58 |
| 3.3 Credit card payments fraud detection use case scenario | 60 |
| 3.3.1 Data science and fraud detection model development | 61 |
| 3.3.2 Fraud detection model deployment and evaluation on IBM zSystems | 66 |
| 3.3.3 Application integrations and considerations | 75 |
| 3.4 Summary | 78 |
| Chapter 4. Other use case scenarios | 81 |

| | |
|--|------------|
| 4.1 The overall picture of your AI use case | 82 |
| 4.1.1 Business-related use case scenarios | 82 |
| 4.1.2 IT-related use case scenarios | 85 |
| 4.2 Detecting CICS anomalies on IBM zSystems scenario | 89 |
| 4.2.1 Background and business goal | 89 |
| 4.2.2 Architecture | 90 |
| 4.2.3 Summary | 91 |
| 4.3 Object detection from geospatial images for cadastral analysis on IBM zSystems | 92 |
| 4.3.1 Background and business goal | 92 |
| 4.3.2 Architecture | 92 |
| 4.3.3 End-to-end testing | 94 |
| 4.3.4 Summary | 94 |
| Chapter 5. Key takeaways | 95 |
| 5.1 Key technologies for AI on IBM zSystems | 96 |
| 5.2 What is next for your AI on IBM zSystems journey | 96 |
| 5.3 Future of AI technologies on IBM zSystems | 98 |
| Appendix A. Installation and configuration pointers | 99 |
| IBM z/OS Container Extensions 2.5 | 100 |
| Red Hat OpenShift Container Platform | 100 |
| IBM Data Virtualization Manager for z/OS | 101 |
| IBM Open Data Analytics for z/OS | 101 |
| IBM Watson Machine Learning for z/OS 2.4.0 | 101 |
| IBM Db2 AI for z/OS 1.5.0 | 102 |
| IBM Db2 for z/OS 13 with SQL Data Insights | 102 |
| AI on IBM zSystems | 103 |
| ONNX resources | 103 |
| TensorFlow resources | 103 |
| IBM CICS Transaction Server for z/OS resources | 103 |
| IBM zSystems Operations Analytics | 104 |
| IBM Cloud Pak for Data | 104 |
| Appendix B. Additional material | 105 |
| Locating the web material | 105 |
| Using the web material | 105 |
| Downloading and extracting the web material | 106 |
| Related publications | 107 |
| IBM Redbooks | 107 |
| Online resources | 108 |
| Help from IBM | 109 |
| Abbreviations and acronyms | 111 |

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](http://www.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 <http://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® | IBM Watson® | Redbooks (logo)  ® |
| Db2® | IBM z13® | SPSS® |
| IBM® | IBM z14® | WebSphere® |
| IBM Cloud® | IBM z16™ | z/OS® |
| IBM Cloud Pak® | Parallel Sysplex® | z13® |
| IBM Research® | Redbooks® | z16™ |

The following terms are trademarks of other companies:

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.

Microsoft, 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.

Red Hat and OpenShift are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

RStudio, and the RStudio logo are registered trademarks of RStudio, Inc.

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

Preface

In today's fast-paced, ever-growing digital world, you face various new and complex business problems. To help resolve these problems, enterprises are embedding artificial intelligence (AI) into their mission-critical business processes and applications to help improve operations, optimize performance, personalize the user experience, and differentiate themselves from the competition.

Furthermore, the use of AI on the IBM® zSystems platform, where your mission-critical transactions, data, and applications are installed, is a key aspect of modernizing business-critical applications while maintaining strict service-level agreements (SLAs) and security requirements. This colocation of data and AI empowers your enterprise to optimally and easily deploy and infuse AI capabilities into your enterprise workloads with the most recent and relevant data available in real time, which enables a more transparent, accurate, and dependable AI experience.

This IBM Redpaper publication introduces and explains AI technologies and hardware optimizations, such as IBM zSystems Integrated Accelerator for AI, and demonstrates how to leverage certain capabilities and components to enable solutions in business-critical use cases, such as fraud detection and credit risk scoring on the platform. Real-time inferencing with AI models, a capability that is critical to certain industries and use cases such as fraud detection, now can be implemented with optimized performance thanks to innovations like IBM zSystems Integrated Accelerator for AI embedded in the Telum chip within IBM z16™.

This publication also describes and demonstrates the implementation and integration of the two end-to-end solutions (fraud detection and credit risk), from developing and training the AI models to deploying the models in an IBM z/OS® V2R5 environment on IBM z16 hardware, and to integrating AI functions into an application, for example an IBM z/OS Customer Information Control System (IBM CICS®) application.

We describe performance optimization recommendations and considerations when leveraging AI technology on the IBM zSystems platform, including optimizations for micro-batching in IBM Watson® Machine Learning for z/OS (WMLz). The benefits that are derived from the solutions also are described in detail, which includes how the open-source AI framework portability of the IBM zSystems platform enables model development and training to be done anywhere, including on IBM zSystems, and the ability to easily integrate to deploy on IBM zSystems for optimal inferencing. You can uncover insights at the transaction level while taking advantage of the speed, depth, and securability of the platform.

This publication is intended for technical specialists, site reliability engineers, architects, system programmers, and systems engineers. Technologies that are covered include TensorFlow Serving, WMLz, IBM Cloud Pak® for Data (CP4D), IBM z/OS Container Extensions (zCX), IBM Customer Information Control System (IBM CICS), Open Neural Network Exchange (ONNX), and IBM Deep Learning Compiler (zDLC).

Authors

This paper was produced by a team of specialists from around the world working at IBM Redbooks, Poughkeepsie Center.

Makenzie Manna is an IBM Redbooks Project Leader in the United States with a primary focus on IBM zSystems software and solutions. She holds a Master of Science degree in Computer Science Software Development from Marist College and a BS in Mathematics from SUNY Cortland. Her areas of interest and expertise include AI technologies and solutions on IBM zSystems, educational course content development, video editing and production, video animation, patent innovation, and technical content development.

Erhan Mengusoglu is a Software Engineer at IBM CICS Development. He is a PhD Software Engineer who is experienced in machine learning (ML), AI, Neural Networks, Complex Event Processing, Model Driven Development, and Speech Recognition domains. In his previous roles, he worked as a developer and tester for IBM Operational Decision Management product, a lecturer (assistant professor) in computer science departments of universities, and a team leader of automated inspection systems development for manufacturing. He also managed several research projects. His programming languages include assembly language (8080), Cobol, Python, and JavaScript.

Artem Minin is a Technical Specialist at the Washington Systems Center specializing in data and AI on IBM zSystems. He focuses on supporting client proof of concepts (PoCs), technology workshops, and implementations of IBM Db2® Data Gate for z/OS, IBM Db2 13 for z/OS SQL Data Insights (SQL DI), WMLz, and Db2 AI for z/OS. Artem graduated with a M.S. in Electrical Engineering from North Carolina State University. As a graduate researcher, Artem developed ML and deep learning (DL) models for agricultural applications, built the software and hardware for Internet of Things (IoT) systems, and embedded AI models into those systems, which were deployed to farms across the US.

Krishna Teja Rekapalli is an Advisory Data Scientist at IBM Sustainability Software at IBM Austin. He holds a master's degree in operations research. As a graduate student, he studied DL, non-linear optimization, and stochastic programming. At IBM, he works on ML and AI models for applications in the areas of Agriculture, Energy & Utilities, and Climate by leveraging geospatial and weather data.

Thomas Rüter is a senior IT Architect for IBM zSystems solutions in the context of Data and AI on IBM zSystems. He holds a graduate degree in physics and teaches machine and deep learning for masters students in engineering at the Black Forrest Campus of the University of Stuttgart. In his current role, he consults IBM clients about the data and AI portfolio on IBM zSystems. In previous roles, he served as an IT Architect in the telecommunication and automotive sectors.

Pia Velazco is a Software Developer in the WMLz Development team. She holds a bachelor's degree in economics. Before joining the WMLz Development team, she worked as an Application Developer and Db2 Systems Programmer, and was part of the IBM Db2 Analytics Accelerator (IDAA) Client Success Team. In her current role, she supports the team's z/OS, CICS, Db2, and WMLz infrastructure.

Markus Wolff is a Technical Specialist for IBM zSystems solutions in the context of data and AI on IBM zSystems. He holds a bachelor's degree in business administration with focus on Industry 4.0 and Digitization. During his studies, he focused on the digitization of business models and the economical parts of IT Operational Analytics (ITOA) and ML. In his current role, he supports clients in PoCs or client workshops in the context of data science and ML regarding WMLz, Db2 AI for z/OS, and CP4D on IBM zSystems.

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

Robert Haimowitz, Patrik Hysky, Lydia Parziale
IBM Redbooks, Poughkeepsie Center

Tom Ambrosio and Bill Lamastro
IBM CPOs

Abid Alam, Joe Bostian, Joy Deng, Christian Jacobi, Suhas Kashyap, James Kuchler,
Nicholas Kunze, Steven Lafalce, Maggie Lin, Abuchi Obiegbu, Andrew Sica,
Elpida Tzortzatos
IBM US

Guillaume Arnould, Stephane Faure, Sebastien Llaurency, Denis Ville
IBM France

Iris Baron and Usman Minhas
IBM Canada

Shuang Yu
IBM China

George Burgess and James O'Grady
IBM UK

Eberhard Hechler, Benedict Holste, Markus Leber, Christian Lenke, Claus Moser,
Khadija Souissi, Rüdiger Stumm
IBM Germany

Kazuaki Ishizaki
IBM Japan

Colleagues from Dataport AöR in Altenholz, Germany

Thanks to the authors of the previous editions of this paper.

- ▶ Authors of the first edition, *Optimized Inferencing and Integration with AI on IBM Z Introduction, Methodology, and Use Cases*, REDP-5661, published in February 2022, were:

Makenzie Manna, Mehmet Cuneyt Goksu, Jens Ole Hein, Erhan Mengusoglu, Erica Ross,
Markus Wolff, Shuang Yu

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 papers to be as helpful as possible. Send us your comments about this paper 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:

<http://www.linkedin.com/groups?home=&gid=2130806>

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

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>

Summary of changes

This section describes the technical changes that were made in this edition of the paper and in previous editions. This edition also might include minor corrections and editorial changes that are not identified.

Summary of Changes

for *Optimized Inferencing and Integration with AI on IBM Z Introduction, Methodology, and Use Cases*, REDP-5661

as created or updated on November 30, 2022.

November 2022, Second Edition

This revision includes the following new and changed information.

New information

- ▶ IBM Db2 for z/OS 13 SQL Data Insights (SQL DI) functions
- ▶ Capabilities comparison between IBM Watson Machine Learning for z/OS (WMLz) 2.4 and WMLz Online Scoring Community Edition (OSCE) and recommendations on when to leverage which solution for optimal results
- ▶ Credit card payments fraud detection use case scenario:
 - Description and business case
 - Data science and fraud detection (long short-term memory (LSTM)) model development details (understanding and preparing the data, selecting the model, data mapping, and training the model)
 - Fraud detection model deployment and evaluation on IBM zSystems (importing and deploying the model with WMLz, Open Neural Network Exchange (ONNX) conversion, reshaping the model for WMLz micro-batching, and testing the model with WMLz)
 - Application integrations and considerations (IBM Customer Information Control System (IBM CICS) program with LINK to ALNSCORE, and CICS program with Representational State Transfer (REST) application programming interface (API))
- ▶ Performance-tuning considerations and recommendations, including micro-batching for better performance

Changed information

- ▶ Added details regarding using artificial intelligence (AI) technology and optimizations on the IBM zSystems platform, such as the IBM zSystems Integrated Accelerator for AI within IBM z16
- ▶ Included feature updates for WMLz 2.4 (including CICS LINK calling for ONNX models)
- ▶ Addition of Db2 AI for z/OS 1.5.0 features
- ▶ Additional information added to the open source on IBM zSystems discussion
- ▶ Planning steps that are required to deploy models to IBM zSystems
- ▶ Additional deployment to IBM zSystems options: ONNX, Predictive Model Markup Language (PMML), and TensorFlow

- ▶ Additional information for inferencing on IBM zSystems options: WMLz, WMLz OSCE, TensorFlow Scoring Server, and IBM zSystems Deep Learning Compiler (zDLC)
- ▶ Cloud Pak for Data (CP4D) 4.5.x updated capabilities



Foundations of artificial intelligence

With much interest in recent years and new use cases emerging everyday, artificial intelligence (AI) can give the illusion of being a new technology. However, the *concept* of AI dates back to ancient Greece and resurfaced in the early 1950s in a paper that was published by Alan Turing: *Computing Machinery and Intelligence*.¹

The excitement around AI took off in the mid-1950s, when the term AI was coined by John McCarthy at the debut AI conference at Dartmouth College. Over a period of several weeks, foundational AI terms and concepts were developed, and later that same year the first ever AI software program was run.

After the inception and a boom phase, the field of AI underwent long periods of stagnation to which AI historians refer to as “AI winters”. Lack of access to affordable high-computing power and data storage are one of the key reasons for these AI winters. As a result, in the period of 1950 - 1990, AI confined itself mostly to research labs. Events in the late 1990s like IBM Deep Blue beating Garry Kasparov in 1997 marked the end of the last AI winter.

Today's implementations of AI systems and applications go much further, encompassing technologies such as natural language processing (NLP), natural language understanding (NLU), and image generation from text and image recognition. When we discuss AI today, we are referring to modern implementations of concepts that date back to research from the mid-20th century.

It is imperative to understand what is meant by and encompassed in the term AI, what drives the development and adoption of AI technologies, and how bringing AI systems onto the IBM zSystems platform can result in business-critical benefits. This chapter introduces you to these concepts and includes the following topics:

- ▶ 1.1, “What is artificial intelligence” on page 2
- ▶ 1.2, “Advantages and value of data and AI on IBM zSystems” on page 4
- ▶ 1.3, “Building and training anywhere and deploying and infusing on IBM zSystems” on page 8

¹ <https://redirect.cs.umbc.edu/courses/471/papers/turing.pdf>

1.1 What is artificial intelligence

According to a 2004 paper that was written by John McCarthy and published by Stanford University, AI is defined as:

“It 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.”²

AI is a field of technology that focuses on building intelligent systems that can perform tasks that can be too complex to confine to a finite set of instructions to a computer. Rather than providing computers with a set of rules, AI techniques enable you to train these intelligent systems by using past data and letting the system learn patterns to solve a problem.

Take the example of a system that recognizes human faces. If you must design a computer program purely based on rules, then you must provide modules to mathematically define different features of a face, like eyes, eye-brows, and a nose. This program also must account for wide variability in features because not every face looks the same. The result is a complex set of rules.

Thanks to AI techniques like deep neural networks (DNNs) and convolutional neural networks (CNNs), any software developer can build a facial recognition program by using a few lines of code and training the model on a few hundred images. The best part about AI techniques is that if the data changes, they can be retrained on the new data and evolve.

There is no question that infusing AI into products and services offers a great deal of competitive advantage to organizations. The digital transformation from the 1960s enabled us to collect, catalog, and store huge amounts of data, but today’s AI can enable organizations to make sense of that data and provide better products and services. The key drivers of high adoption of AI in the last few decades can be broken down into two technical influential factors: large amounts of usable data, and increased access to cheap computing power.

Thanks to all the recent advances, today AI is everywhere: It is in your cellphone as a speech-to-text feature in messaging, or speech recognition to perform voice searches with AI Assistants like Apple Siri and IBM Watson Assistant for businesses. Most businesses today are incorporating AI into their customer service experience through chat bots and virtual agents.

Corporations like CVS Health are leveraging Watson Assistant to effectively handle a 10-fold spike in call volume for COVID-19 vaccines.³ Additionally, McDonalds worked with IBM to improve their customer’s drive-through experience by using AI.⁴ These partnerships are a few of the many examples of how enterprises are leveraging AI to improve processes and experiences for themselves and their customers.

AI techniques like neural networks are data hungry and require huge amounts of data to train, test, and deploy a usable and trustworthy model of acceptable quality. Along with the key drivers of affordable storage and easy access to computing power, cloud computing is another key technology that makes AI more accessible by enabling anyone with access to the internet to build, train, and deploy complex AI models.

² https://borghese.di.unimi.it/Teaching/AdvancedIntelligentSystems/01d/IntelligentSystems_2008_2009/01d/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatissai.pdf

³ <https://www.linkedin.com/pulse/fueling-cvs-healths-covid-response-ai-shobhit-varshney/>

⁴ <https://newsroom.ibm.com/Joint-Statement-from-McDonalds-and-IBM>

Today, AI has come far from being a research domain, and its use cases span many mission-critical areas for businesses, including better customer experience, cybersecurity, and improved operational efficiency.

As part of the IBM Hybrid Cloud and AI strategy, huge investments were made in infusing AI into IBM products, and bringing modern AI capabilities to IBM zSystems is at the core of that strategy. Before we explain the value and capability of AI on IBM zSystems, we must describe the differences in the terminology.

AI is a term that encompasses subdisciplines, such as machine learning (ML) and deep learning (DL). These terms are often used interchangeably, but it is important to understand the nuances that differentiate them from each other.

1.1.1 Artificial intelligence, machine learning, and deep learning

When discussing AI, ML, and DL, it is important to understand the differences between each of these terms. Figure 1-1 shows the key differences between these terms and the inclusion hierarchy into which the terms fall.

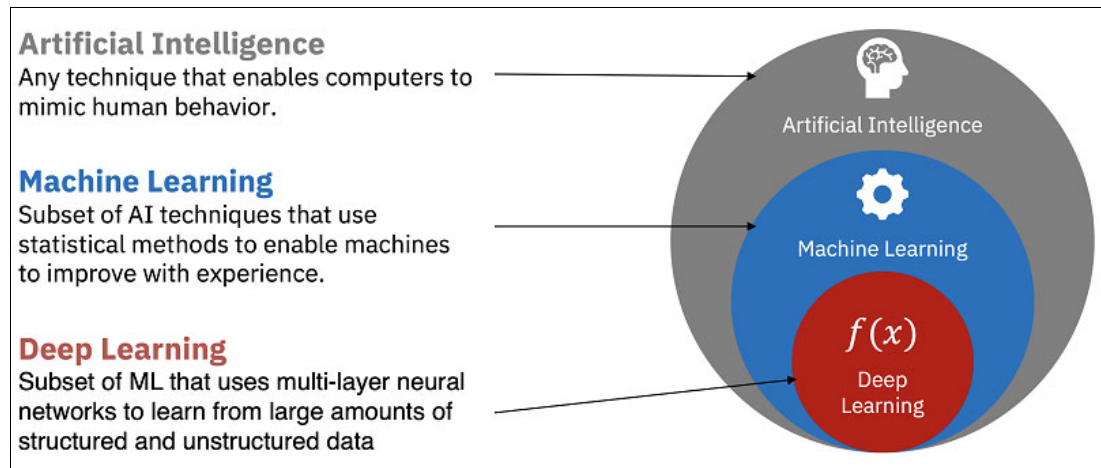


Figure 1-1 Differences between AI, ML, and DL

The term *AI* refers to techniques that embody the purpose of mimicking human behavior. The purpose is to enable computers to perform tasks that are regarded as uniquely human, that is, processes that require intelligence. However, for certain problem classes, AI algorithms surpass humans.

ML is a subfield of AI techniques that is sometimes referred to as *classical* or *nondeep AI*. ML focuses on using a certain class of algorithms and statistical methods for problem solving. Although any AI technique is only as good as the quality of the data on which it is trained, ML techniques depend heavily on features. *Features* represent attributes that are used as input for the model to train. For that reason, ML algorithms typically require a structured data set and have a high emphasis on feature engineering. Algorithms like logistic regression, decision trees, random forests, and support vector machines fall under ML techniques. Use cases that typically involve ML techniques include data insight and prediction from structured data sets.

DL is yet another subfield of AI techniques that is a subfield within ML. DL techniques build on ML techniques with more complex algorithms, and focus on the usage of neural networks for problem solving.

As implied in their name, neural networks are based on the biological neural networks of the human brain and are structured to mimic its behavior. However, the latest innovations, such as DNNs, CNNs, recurrent neural networks (RNNs), and transformers have evolved, and they have little to no commonality with biological neural networks.

As opposed to ML techniques, DL techniques need little feature engineering and need huge amounts of data to train. Because they require minimal feature engineering, DL models are sometimes referred as “black-box” models. As a result, a great deal of recent research focuses on the explainability of DL models. Where a typical ML model has hundreds of parameters, some modern DL models have billions of parameters. DL techniques work well when the problems involve unstructured data like text, images, audio, and video, which is why DL techniques outperform ML techniques when it comes to complex problems such as image object detection, NLP, and speech recognition.

1.2 Advantages and value of data and AI on IBM zSystems

Now that you have a better understanding of the foundation of AI and its subdisciplines, we move on to AI on IBM zSystems. This section guides you through some of the different options that are available on IBM zSystems for AI solutions; why IBM zSystems is the platform of choice for AI; and how integrating AI on IBM zSystems with other platforms and tools (including open source) is easier than before and continuously expanding.

1.2.1 Delivering innovation with AI and IBM z16

When seeking to understand the different offerings that are available on IBM zSystems in the context of data and AI, it is helpful to map the offerings into the following architectural layers:

- ▶ Data layer
- ▶ Analytics Engine layer
- ▶ AI platform layer
- ▶ AI solutions layer

Each of these layers has specific product offerings and solutions to help enable data and AI capabilities on the IBM zSystems platform. Each layer, and everything that layer encompasses, is critical to the success of your AI implementation on any platform.

Data layer

The data layer includes data source offerings, such as databases. These data sources can be on the IBM zSystems platform or on an alternative platform, for example, an x86 platform. In either case, offerings such as IBM Data Virtualization Manager for z/OS (DVM) offer data access to various different data sources, enabling you to accomplish the following tasks:

- ▶ Combine and integrate non-relational and relational data.
- ▶ Integrate IBM zSystems and non IBM zSystems data.
- ▶ Incorporate unstructured data with structured data.
- ▶ Provide SQL access to non-relational data.
- ▶ Modernize mainframe applications and enhance non-mainframe applications with mainframe data.

For the complete list of DVM for z/OS supported data sources and types and other technical details, see [Supported data sources](#).

Analytics Engine layer

This layer includes offerings that are commonly referred to as the “engine” for analytics and AI on the IBM zSystems platform. One of the Analytics Engine solutions on IBM zSystems is IBM Open Data Analytics for z/OS (IzODA). IzODA is a foundation for ML, and it offers Spark and Anaconda run times for software solutions, such as IBM Watson Machine Learning for z/OS (WMLz). Additionally, IzODA provides an optimized data layer that serves as a common interface for analytics applications to access z/OS data sources, such as IBM Db2, VSAM, and SMF.

For more information about IzODA, see [IBM Open Data Analytics for z/OS](#).

In addition to the ML engine, various open source images and AI compiler images on IBM zSystems can be accessed at the [IBM zSystems and LinuxONE Container Registry](#), which is described in “IBM zSystems and LinuxONE Container Registry” on page 8.

AI platform layer

Included in the AI platform layer are offerings that provide enterprise-ready capabilities to develop, train, score, and retrain ML and DL models. This layer also includes the scoring and inferencing capability of AI models. WMLz is an example of a such an offering in this layer that enables organizations to place ML and DL models on z/OS, where transactions originate for IBM zSystems customers.

Aside from offering model development and lifecycle management, WMLz primarily focuses on deploying and inferencing AI models closer to where business-critical transactions occur. This function helps ensure real-time scoring with optimal performance and scalability while keeping business-critical data secure by using IBM zSystems strengths like the IBM z16 Integrated Accelerator for AI. To improve the ease of adoption and integration, WMLz supports many AI frameworks that are widely adopted in the data science world, and it supports standard exchange formats such as Predictive Model Markup Language (PMML) and Open Neural Network Exchange (ONNX).⁵

For more information about WMLz, see [IBM Watson Machine Learning for z/OS](#).

IBM Cloud Pak for Data (CP4D) is another offering that is instrumental to the future of AI solutions on IBM zSystems. CP4D runs on the IBM zSystems infrastructure in addition to other platforms based on Red Hat OpenShift, and it is deeply integrated with WMLz. Typical use cases involve performing model development, training, and bias checks on CP4D, and the model deployment and scoring on WMLz.

For more information about the latest release of CP4D, see [IBM Cloud Pak for Data](#).

For more information about potential use cases that leverage the strengths of the IBM zSystems infrastructure with the capabilities that are offered by CP4D, see [Capabilities on Linux on IBM zSystems and IBM LinuxONE](#).

AI solutions layer

The AI solutions layer includes “black-box” solutions that use AI for a specific purpose or to solve a specific problem. Such solutions are designed to help automate the more complex ML and data science processes by infusing AI throughout an organization and into business processes and applications. These solutions are referred to as “black-box” solutions because they transform these difficult to understand, complex tasks and processes into simple, actionable steps without requiring a specific skillset, and provide clarity and understanding around the decisions that are made by a model.

⁵ <https://www.ibm.com/docs/en/wml-for-zos/2.4.0?topic=owz-supported-algorithms-data-sources-data-types-model-types>

Solutions in the area of IT Operational Analytics (ITOA), such as IBM Db2 AI for z/OS (Db2ZAI) or IBM zSystems Operations Analytics (IZOA) are built on and powered by the ML services that are provided by WMLz to help optimize operational performance and maintain system and subsystem health by using deep, AI-driven insights.

IBM Db2 13 for z/OS SQL Data Insights (SQL DI) is an excellent example of an AI-driven “black-box” solution for business insights. SQL DI uses DL and IBM zSystems technology to enable the industry first ability to infer hidden information within Structured Query Language (SQL) queries in a relational database by detecting and extracting semantic similarities between two records to enhance traditional data processing. For more information about SQL DI, see “IBM Db2 13 SQL Data Insights” on page 84.

For more information about Db2ZAI, SQL DI, and IZOA, see the following websites:

- ▶ [IBM Db2 AI for z/OS](#)
- ▶ [Running AI SQL Queries with SQL Data Insights](#)
- ▶ [IBM zSystems Operations Analytics overview](#)

Other AI solutions on the IBM zSystems platform use different architectural approaches, for example, TensorFlow. For more information about the use of container technology, see 3.2, “Credit risk assessment use case scenario” on page 48. This section features a use case in which TensorFlow and WMLz (both on the IBM zSystems platform) are integrated to score a DL model.

Why AI on IBM zSystems

One of the many benefits of implementing your AI solution on IBM zSystems is the broad range of offerings that are available on the platform to support each stage of your data and AI solution. The IBM zSystems infrastructure also offers a robust, secure, and scalable infrastructure for enterprise applications and transactions.

Because most business transactions of large enterprises are processed on IBM zSystems, performing analytics and AI predictions on the platform positions those processes close to the source of the transaction and data. This phenomenon of moving AI applications and frameworks close to the business data is called *data gravity*. Data gravity enables real-time analytics, leaves sensitive data in a secure data environment, and reduces latency and network traffic because the need for data movement is removed.

IBM is making substantial investments in ensuring that IBM zSystems is a highly optimized platform for AI. Developments such as the IBM Telum processor chip with the IBM zSystems Integrated Accelerator for AI are positioning IBM as the premier platform provider for inference workloads. This hardware innovation speaks to the commitment and focus IBM made to support and enable AI and improve inferencing at scale.

For more information about the IBM zSystems Integrated Accelerator for AI on z16, see the following resources:

- ▶ [IBM z16 \(3931\) Technical Guide, SG24-8951](#)
- ▶ [IBM Telum Processor: the next-gen microprocessor for IBM zSystems and IBM LinuxONE](#)
- ▶ [IBM z16 puts innovation to work while unlocking the potential of your hybrid cloud transformation](#)

For more information about use cases for the IBM zSystems Integrated Accelerator for AI, see 5.1, “Key technologies for AI on IBM zSystems” on page 96.

1.2.2 Open-source frameworks on IBM zSystems

Together, IBM and Red Hat are one of the top three organizations with the most active open-source contributors and over 6,300 combined active contributors in 2022 as of September, according to the Open Source Contributor Index.⁶

This commitment to open-source enablement is not a new trend for IBM. The enterprise was one of the earliest champions of open source, dating back to the 1990s with contributions to Linux. Some of the most notable IBM open-source contributions over the years include contributions to the Linux kernel; Java: open-source cloud projects including Cloud Foundry; open-source container projects including Docker and the Open Container initiative; the blockchain Hyperledger project; Node.js; JavaScript, and quantum projects.

IBM is heavily invested in working with public communities to bring the best of the open-source world to IBM zSystems. There is a sustained focus on expanding the availability of data science and ML capabilities on the platform by offering many open-source frameworks on Linux on IBM zSystems and z/OS natively. To support this focused effort, IBM created the [IBM Center for Open-source Data and AI Technologies](#).

In addition to formal offerings that bring great value to AI capabilities on IBM zSystems, the best of the open-source community can be leveraged on the platform. IBM is an active contributor to the open-source AI community, and it contributes to projects such as TensorFlow, Kubeflow, Project Jupyter, Apache Spark, and PyTorch.

Some of the readily available AI frameworks for use on the IBM zSystems platform include scikit-learn, XGBoost, Apache Spark, PyTorch, and TensorFlow.

With TensorFlow being one of the most widely used open-source frameworks for ML and DL, it was critical for IBM to enable the use of the TensorFlow framework on the IBM zSystems infrastructure. For more information about how IBM enabled the TensorFlow framework on IBM zSystems, see 1.3, “Building and training anywhere and deploying and infusing on IBM zSystems” on page 8.

PyTorch is another DL open-source framework that was widely adopted by enterprises because of its simplicity, flexibility, and modularity. Developers and data scientists can use PyTorch to create or import AI applications on the IBM zSystems platform in a simple and intuitive way and easily integrate with other Python libraries. You can use PyTorch on the platform by building your own PyTorch package or by using the Anaconda support package for IBM zSystems.

Anaconda is available on IBM zSystems with various supported packages with which data scientists can build, import, and test open-source solutions on the IBM zSystems platform while still working with their choice of tools and frameworks, such as scikit-learn, PyTorch, and Jupyter. Because Anaconda runs natively on Linux on IBM zSystems and through IBM z/OS Container Extensions (zCX) in z/OS, your AI solution can leverage and benefit from the data gravity of the platform. To access a full list of Anaconda on IBM zSystems supported packages, see [Packages for 64-bit Linux on IBM zSystems CPUs with Python 3.8](#).

Note: TensorFlow and PyTorch can both be run as container images in zCX or in Linux on IBM zSystems, as described in “IBM zSystems and LinuxONE Container Registry” on page 8.

⁶ <https://opensourceindex.io/>

Enabling the use of these popular open-source AI frameworks on IBM zSystems is a critical element in bringing AI to where your mission-critical applications and data are kept. Collocating your data and applications on IBM zSystems and infusing them with AI while leveraging the platform's strengths, such as supporting the use of the IBM Integrated Accelerator for AI in the z16, allows your business to gain real-time insights and unlock new business value.

IBM zSystems and LinuxONE Container Registry

On IBM zSystems, popular AI frameworks are available as Linux-based container images, which can be used to create containers in zCX or Linux on IBM zSystems. These container images consist of executable application code, tools, libraries, and required dependencies for the framework to run in a container. Images can be reused as templates for building custom containers, so it is possible to create an environment fitting your development needs.

Although possible, most developers do not build container images from scratch. Instead, they are generally pulled from a private or trusted public repository, such as the IBM Cloud® Container Registry, which provides a multi-tenant private image registry that is highly available (HA), scalable, and encrypted to store, access, and manage your containers. You can also use IBM Cloud Container Registry to access public images that are provided by IBM.

The IBM zSystems Container Image Registry is a mechanism for obtaining prebuilt container images of popular open-source software. The images in this repository are built from source, optimized for s390x architecture, build-tested, and scanned for known vulnerabilities. Several popular AI frameworks such as PyTorch, TensorFlow, and TensorFlow Serving are available as container images in the repository. You can access the registry by going to [IBM zSystems and LinuxONE Container Registry](#).

The open-source system IBM supports is a vast and growing resource that elevates all the features that IBM zSystems offers. For more information about open-source packages that are portable or validated on corresponding distribution versions by IBM, see [IBM zSystems and LinuxONE Community Validated Open Source Software](#).

1.3 Building and training anywhere and deploying and infusing on IBM zSystems

The ability to build and train AI models anywhere, deploy them on IBM zSystems, and infuse models into workload applications with seamless optimization adds value to the platform. Such an ability allows enterprises to maximize their investments in AI platforms and for data scientists to use the framework that best suits their skill set.

Taking a model, deploying it on IBM zSystems, and infusing models into workload applications while benefiting from the platform's unparalleled qualities of service is valuable for an organization that hopes to reliably leverage business-critical insights that are gained from the model and the data that it is powered by. This section introduces the DL ecosystem on IBM zSystems and the process to deploy models on the platform.

Figure 1-2 on page 9 illustrates the process that an enterprise can follow to deploy an AI model on IBM zSystems. Start with building and training a model on any public cloud or on-premises infrastructure by using a preferred model framework. Then, easily deploy that model on IBM z16 to seamlessly take advantage of innovative hardware optimizations, such as the IBM z16 Integrated Accelerator for AI, to infuse AI into every transaction.

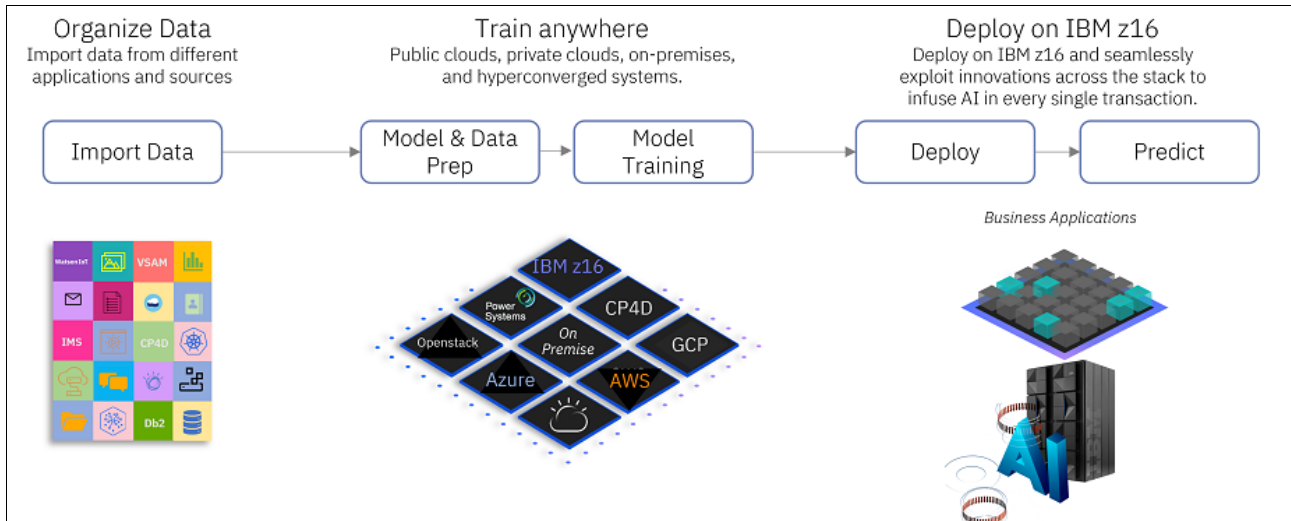


Figure 1-2 Training anywhere and then deploying on IBM zSystems

Data that is specific to the business problem that must be addressed must be organized from various sources. Depending on the AI model that is built, this data might be images, text, audio, database tables, VSAM data sets, SMF data, or a table from a data warehouse. It is important to be cognizant of the data sources that are used to build and train the model because they must be present at the time of inferencing. If you are leveraging the data gravity of the IBM zSystems platform by using model input data that is on the platform, then you minimize your inferencing latencies.

A DL or ML model, depending on the business challenge, must be built and trained by data scientists with the support of people that possess domain knowledge. The development and the training are done by using a framework that is familiar to the data scientist and can be completed on IBM zSystems or any another platform. With this approach, you can maximize existing or future infrastructure investments.

Figure 1-3 shows several supported AI frameworks and libraries that can be used to import many developed models into various runtime environments on IBM zSystems, 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 zSystems platform seamlessly into their native runtime environment, which typically refers to an Anaconda, Python, or Spark environment in Linux on IBM zSystems or zCX.

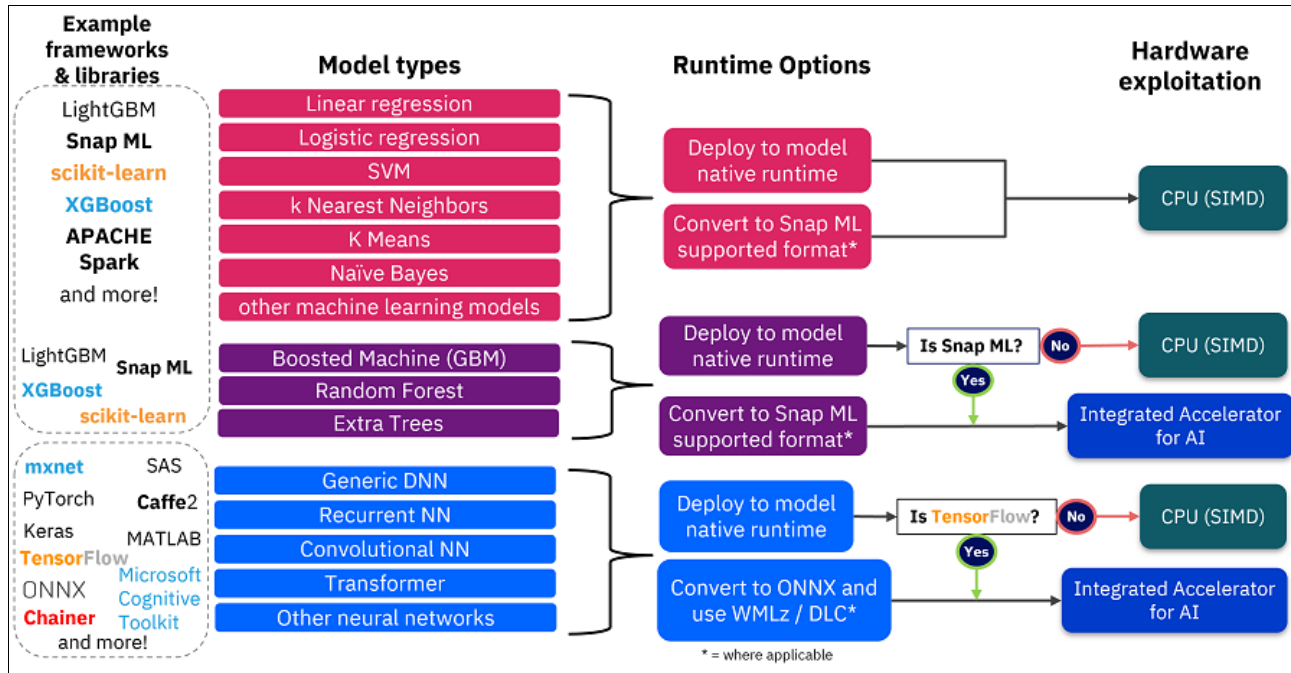


Figure 1-3 AI frameworks on IBM zSystems

To import the model to the IBM zSystems platform, you can use the Linux Secure Copy Protocol (SCP) to transfer the model between two different systems by using the SSH protocol. For more information about deployment options, see 1.3.1, “Modernizing z/OS applications by infusing AI” on page 12.

Some classical ML model types, such as gradient boosting machine, random forest, and extra trees, can take advantage of the IBM zSystems 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](#).

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.

In this example notebook that is provided by IBM on GitHub, [Example_credit_card_fraud.ipynb](#), a random forest model that was converted to the Snap ML random forest model format experienced an inferencing speedup of 26 times. This speed-up result depends on the model and data that are used, so it is a best practice to compare inferencing speeds for your specific use case. During training or conversion, you can decide whether to run the model on a CPU or use the IBM zSystems Integrated Accelerator for AI.

Disclaimer: Performance results always depend on the hardware and software environment. For more information about the environment that was used to run this notebook, see [Example_credit_card_fraud.ipynb](#).

DL models that are built with AI frameworks such as TensorFlow, PyTorch, mxnet, SAS, Caffe2, MATLAB, Microsoft Cognitive Toolkit, and several others support conversion to the ONNX format. ONNX is an open and platform-neutral format for describing AI models that facilitates the production deployment of models independent from their training environment and framework. For more information about the ONNX open-source project, see [Open Neural Network Exchange](#).

After the model is converted to an ONNX format, it can be compiled by using the IBM zSystems Deep Learning Compiler (zDLC) to take advantage of the IBM z16 Integrated Accelerator for AI. You can use the zDLC for ONNX model compilation automatically within WMLz or manually within a container image of the zDLC. The Integrated Accelerator for AI is engineered for accelerating the processing of several complex mathematical operations that typically are found within DL models. As of IBM z16, the Integrated Accelerator for AI supports the following mathematical operations that are found within models:

- ▶ Long short-term memory (LSTM) activation
- ▶ GRU Activation
- ▶ Fused Matrix Multiply, Bias op
- ▶ Fused Matrix Multiply (with broadcast)
- ▶ Batch Normalization
- ▶ Fused Convolution, Bias Add, ReLU
- ▶ Max Pool 2D
- ▶ Average Pool 2D
- ▶ Softmax
- ▶ ReLU
- ▶ Tanh
- ▶ Sigmoid
- ▶ Add
- ▶ Subtract
- ▶ Multiply
- ▶ Divide
- ▶ Min
- ▶ Max
- ▶ Log

TensorFlow models can be imported to a native model run time on IBM zSystems, but you must account for several considerations. For example, if the TensorFlow model was trained on an s390x architecture in a TensorFlow release before Version 2.7, then the memory byte order (endianess) of the model is the opposite of the format that is expected by IBM zSystems. The model will not be transferred correctly, and you must use a utility to switch the endianess of the TensorFlow model's protobuf graph. You can access the endian converter at [Endian_convertor](#).

Note: TensorFlow models that are trained in TensorFlow 2.7 or later can be imported into IBM zSystems without endian conversion. If any issues are encountered, contact the AI on IBM zSystems team by sending an email to aionz@us.ibm.com.

TensorFlow models that are imported with this method *cannot* leverage the IBM z16 Integrated Accelerator for AI.

Therefore, IBM developed a method for inferencing a TensorFlow model on IBM z16 while taking advantage of the Integrated Accelerator for AI. Available as an open-beta program, IBM provides a TensorFlow container image in the IBM zSystems Container Registry that is named *IBM zSystems Optimized for TensorFlow*, which supported native runtime acceleration within a container. TensorFlow models can be built and trained on any platform with TensorFlow core 2.7 and deployed to this container. When the model performs inferencing operations within the container, the TensorFlow graph execution transparently targeted the Integrated Accelerator for AI, which requires no changes to the model. You can toggle between using the CPU and the Integrated Accelerator for AI for inferencing requests by modifying an environmental variable in the container, which you *might* want to do for testing purposes.

Note: With this option, inferencing is accelerated only within the container itself, which means that predictions must be driven from within the container. There is no Representational State Transfer (REST) application programming interface (API) function like with TensorFlow Serving, so you must custom-build that function.

AI offerings for IBM zSystems increasingly support open-source technology and open standards, as described in 1.2.2, “Open-source frameworks on IBM zSystems” on page 7. This support allows an exchange of assets within IBM zSystems and with other platforms. Models can be developed and trained anywhere, that is, on any infrastructure and with any supported tool.

In the next section, we explore different deployment options that are present on IBM zSystems.

1.3.1 Modernizing z/OS applications by infusing AI

Before we describe the different options and offerings that enable AI on IBM zSystems, you must understand the deployment options that are available on the platform. In the context of this publication, *deployment* refers to the process of configuring an AI model to integrate with other applications or users on IBM zSystems.

Traditionally, when you think of running applications and services on the IBM zSystems platform, you think of doing so with the z/OS operating system. Being an operating system, z/OS offers significant benefits, including a secure environment for high availability (HA) enterprise workloads.

Containerized applications

z/OS has zCX, which enables you to run Linux based containers through a Docker interface within a z/OS environment. This capability allows the deployment of Linux on IBM zSystems applications as Linux-based containers within a z/OS system to directly support workloads with an affinity to z/OS. No special IBM zSystems skills are needed to manage the Linux workloads running in zCX, so existing Linux development, administration, and container skills can easily be leveraged.

Also, running these containerized applications on z/OS enables you to streamline your DevOps environment and retain the benefits of z/OS qualities of service, such as security and HA. With zCX, applications' resources can be managed by z/OS Workload Manager and disaster recovery (DR) can be provided by leveraging an IBM Parallel Sysplex®.

Managing and running containerized applications by using Red Hat OpenShift Container Platform

Complementing these benefits is the ability to use the Red Hat OpenShift Container Platform on IBM zSystems. Red Hat OpenShift Container Platform is a platform that is used to manage and run containerized applications. It enables safe software deployment by automating and securing application container orchestration and lifecycle management within its cluster.

Red Hat OpenShift Container Platform helps position IBM zSystems to be an essential and optimal environment for hybrid cloud *and* traditional enterprise computing. For more information about Red Hat OpenShift Container Platform on IBM zSystems, see [Red Hat OpenShift Container Platform on IBM zSystems and LinuxONE: Reference Architecture](#).

Collocating containerized applications with traditional workloads on IBM zSystems

The ability to colocate containerized applications, including cloud-native applications, with traditional workloads that are running in Linux on IBM zSystems or z/OS brings the AI application close to the data and introduces the advantage of reduced inferencing latency. AI models can be deployed in z/OS container extensions, Red Hat OpenShift Container Platform, and natively within z/OS.

AI deployment solutions on IBM zSystems

There are several solutions that built for deploying AI models on IBM zSystems, such as the following ones:

- ▶ Deploying with IBM Watson Machine Learning for z/OS
- ▶ Deploying with WMLz Online Scoring Community Edition
- ▶ Deploying with TensorFlow Serving
- ▶ Deploying with IBM zSystems Deep Learning Compiler

The first three solutions that are listed above provide several useful deployment features, such as model versioning, RESTful API scoring functions, and high scalability. To leverage the Integrated Accelerator for AI within custom-built applications or scoring services, an enterprise can use the IBM zDLC to compile an ONNX model into an executable shared object library. Additionally, models can be used for inferencing within their native Python or Anaconda runtime environment for use by custom applications within a Linux on IBM zSystems, or zCX environment.

Note: Using this methodology requires development effort if you want support for RESTful API scoring calls from z/OS applications. Additionally, other enterprise-grade model serving features must be created.

Deploying with IBM Watson Machine Learning for z/OS

WMLz is an enterprise-grade, production-ready, ML platform that enables embedding ML and DL models into transactional applications for real-time insights. With WMLz, organizations can build, deploy, and run models on IBM zSystems and leverage several essential enterprise-grade ML features, such as model versioning, auditing, and monitoring.

For z/OS users, WMLz is a good-fit platform for deploying models that were trained on other platforms and converted to a PMML (an XML-based file format for describing ML models and pipelines) or the ONNX model format. Many ML frameworks and tools, such as scikit-learn, R Studio, IBM SPSS® Modeler, and SAS, support converting and persisting models in PMML format.

ONNX is a standard format that also is used for ML models that is especially versatile at representing many DL models that are built in various frameworks. ONNX defines a set of common operators and a common file format that enables the portability of DL models across frameworks and platforms. Several ML frameworks, such as TensorFlow, PyTorch, and scikit-learn, support conversion to the ONNX format.

Additionally, WMLz is designed for HA, strong security, high performance, and low inference latency. It supports model deployment monitoring through IBM Watson OpenScale in CP4D. With Watson OpenScale, an enterprise can analyze AI model deployments to transparently understand and explain why an AI model is making certain predictions. In addition, deployments can be monitored for bias and accuracy drift, enabling trustworthy AI practices.

When an ONNX model is imported in WMLz and deployed in WMLz, the zDLC is automatically used to optimize the model for the IBM zSystems Integrated Accelerator for AI. When the model is optimized for the platform, it can be deployed to an online scoring service, which allows any z/OS applications to obtain inferencing predictions by using a RESTful API call. Alternatively, an option that might appeal to enterprises that want to embed a model within an IBM Customer Information Control System (IBM CICS) transaction is the ability to deploy a model directly to a CICS region. With this option, inferencing calls can be triggered by using CICS **GET** and **PUT** container commands versus doing a RESTful API call.

For more information about model interoperability and technical details regarding the process of the zDLC and WMLz integration to deploy a model, see 2.2, “Integration and deployment best practices” on page 24.

Deploying with WMLz Online Scoring Community Edition

The WMLz Online Scoring Community Edition (OSCE) is a special, no-charge version of WMLz that is intended for simple, non-production testing of the real-time scoring function of pretrained ONNX models in WMLz. WMLz OSCE is packaged as an s390x Docker container image that is easily deployed in a zCX. WMLz OSCE can be used for rapid use case evaluation of embedding DL models in transactional z/OS applications while leveraging the Integrated Accelerator for AI. You can download the trial code for WMLz OSCE at [IBM Watson Machine Learning for z/OS](#).

Deploying with TensorFlow Serving

TensorFlow Serving is an open-source, high-performance deployment option that is a good fit for enterprises that are heavily invested in the TensorFlow ecosystem or have complex model pipelines.

TensorFlow Serving is available as a container image in the IBM zSystems Container Image Registry, and it can be used in a zCX or a Linux on IBM zSystems environment. Any z/OS application can access the TensorFlow model by using a REST API call. The TensorFlow model that is deployed in the TensorFlow Serving server cannot leverage the IBM z16 Integrated Accelerator for AI.

Another TensorFlow deployment option is available in the form of a container image in the IBM zSystems Container registry. This option is named *IBM zSystems Optimized for TensorFlow*. TensorFlow models can be built and trained on any platform with TensorFlow core 2.7, and then deployed to this container. When the model performs inferencing operations within the container, the TensorFlow graph execution transparently targets the Integrated Accelerator for AI with no changes being required to the model. With this option, inferencing is accelerated only within the container itself, meaning that predictions must be driven from within the container running in a zCX or Linux on IBM zSystems environment. There is no REST API function, so inferencing requests from z/OS applications are not possible without more development.

Deploying with IBM zSystems Deep Learning Compiler

Another open-source deployment option that is available on the platform is by using the zDLC container image to compile and optimize ONNX models. The zDLC is available as a container image in the IBM zSystems Container Image Registry, and it can be used in a zCX or a Linux on IBM zSystems environment to create custom applications.

You can use the zDLC to compile an ONNX model to a shared object library that can be leveraged by C, C++, Java, and Python applications to take advantage of the IBM zSystems Integrated Accelerator for AI. Custom scoring services can be created, or new applications that require optimized inferencing can be built.

Note: Any RESTful API functions or z/OS interoperability must be developed.

For more information about or to access the zDLC container images, see [IBM zDLC GitHub](#).

IBM continues to extend the capabilities of the IBM zSystems platform. Broadening the integration capabilities for the platform enables diverse AI development and deployment processes. For more information about the future of AI technology on IBM zSystems, see 5.3, “Future of AI technologies on IBM zSystems” on page 98.



Methodology and tools

Machine learning (ML) attempts to solve business problems that can be resolved by discovering specific patterns in available data. It uses different types of algorithms that learn and improve through data versus programming.

In this chapter, we describe the methodology and tools for IBM zSystems for ML projects. We introduce the methods and processes, and describe the different considerations and components of the tools that are involved in each process, including recommended architectures for your first ML project.

This chapter includes the following topics:

- ▶ 2.1, “Developing a machine learning model” on page 18
 - Defining the business problem
 - Approaches and selecting an approach
 - Machine learning lifecycle
- ▶ 2.2, “Integration and deployment best practices” on page 24
 - Model interoperability
 - Piloting your first project
 - IBM Watson Machine Learning for z/OS overview and integration points
 - Batch versus online inference
 - Continuous integration and continuous delivery in ML

2.1 Developing a machine learning model

An ML model is the result of training an ML algorithm with data over time. As the algorithms import the training data, the model accuracy improves and produces a more precise model that is based on that data, which in turn provides better business value.

However, model accuracy is not solely dependent upon how many times you iterate through the training process. It is also heavily dependent upon the quality and quantity of the data that is being used to train the model. Bad training data can lead to misinterpreting trends, bias, not recognizing new patterns, unexplainable model responses, and more. The training data must be large enough in quantity and be refined, accurate, consistent, and meaningful.

Before getting too caught up in the exciting benefits you can achieve with implementing an ML model, it is important for the success of your artificial intelligence (AI) project to ensure that your expectations are reasonably set, and that a plan is established.

2.1.1 Defining the business problem

At the beginning of every AI project exists a business problem that you must solve. To solve a problem, you first must clearly define the exact problem. Having a good understanding of the nature of the business problem at hand is a critical component to successfully apply ML techniques to solve the problem and support business strategy. For example, a sudden increase in fraudulent activity or unexplainable performance issues that are hurting your core business.

Another important consideration to make when you define and try to understand the business problem is what data sources are available. The use of untapped data sources within your enterprise can help uncover hidden insights. Taking advantage of these sources is critical to gain a better understanding of threats and opportunities for your business goals.

After you have a clear idea of your problem, you can attempt to find a solution. This process can take many forms, from classifying transactions and blocking them if you suspect fraudulent behavior, to analyzing vast amounts of data in batches to make predictions.

Best-for-fit solutions often depend on the industry in which you operate, the type of data you can access, and the results that you want to achieve. Along with establishing what the solution should look like, you must establish parameters for success. To help establish these parameters, consider asking questions that are similar to the following examples:

- ▶ How accurate does the model need to be?
- ▶ How will you measure the effect of the model?
- ▶ What are the performance targets?

Depending on your business and the problem you are addressing, ethical, and regulatory concerns or requirements might need to be addressed. Although outside the scope of this IBM Redbooks publication, these ethical and regulatory concerns and requirements are an important consideration.

2.1.2 Approaches and selecting an approach

As part of your model, you must define the inputs and the wanted output. The inputs can be as easy as the fields of a loan application form, or the performance statistics that are obtained through some type of monitoring interface.

However, perhaps the inputs must be supplemented by historical data, or other data sources that are available to you. For training the model, you need a larger number of examples of this input data to shape the model to your business needs. The type of data that you are working with affects the approach that you take to training and developing your model.

One important item to consider is whether your model requires labeled data. Labeled data describes raw data that includes labels to identify that data and specify its context for the model. When working with labeled data, you can create a training data set that contains labeled input and output data and use that information to train the model and obtain accurate predictions.

For example, your data set contains customer's inputs to a loan application form, including whether they defaulted on the loan. You use that data set to train the model through iterative prediction making, measuring its own accuracy by comparing it to the correct output, and adjusting for the correct answer improving its accuracy. This approach to ML is called *supervised learning*.

When you are setting up your training data, creating labeled data requires domain expertise, more effort, and more time because the value of the label must be expertly and accurately assigned. You must establish the value of your label for each sample in your training data set. Depending on the problem, this process might require human intervention, which results in a higher likelihood for human error.

For example, if you were to classify pictures of animals, you first need a human to correctly label all pictures before you can train your model. However, the use of a supervised learning approach to your ML algorithm can offer a valuable solution to eliminate manual classification work and to make predictions that are based on labeled data.

For unlabeled data, we use an approach called *unsupervised learning*. Unsupervised learning uses ML algorithms to find groupings, or hidden patterns, in the data without the need for human intervention. An example is a recommendation engine that suggests add-on items to customers in an online shop, based on the purchases of similar customers.

Because you do not need to label the data in unsupervised learning, it does not require human intervention at the beginning of the process. Therefore, you can start training the model sooner.

However, the model still needs human intervention because the interpretation of the patterns it discovers is not part of the model. Also, you might discover that some of the insights that are uncovered by your model might not be of value.

Building on our example, an online shop might notice that diapers are often purchased together with red wine; however, it might not make sense to recommend the cross-selling of diapers and red wine. This distinction requires human intervention.

But what if labeled and unlabeled data exists in your data set and labeling that data is too costly? When this issue occurs, we use an approach that is called *semi-supervised learning*. Semi-supervised learning uses a smaller labeled data set for the supervised portion of the training. It also performs the feature extraction through unsupervised learning of a larger unlabeled data set.

Unsupervised learning is a valuable approach to explore masses of data that you want to analyze, but are not sure what exactly it is you are looking for. The goal for unsupervised learning is to get valuable insights from large volumes of new data. Supervised learning is the right approach if you know what you are looking for and must predict outcomes for new data.

Other approaches also are available, such as reinforced learning that requires no data in advance, and semi-supervised learning. For simplicity, we focus here on supervised and unsupervised learning.

2.1.3 Machine learning lifecycle

The ML lifecycle is a multi-phase, partially iterative cycle to ensure trustworthy, explainable, consistent, and reliable predictions (see Figure 2-1). Each phase includes specific processes that are described next.

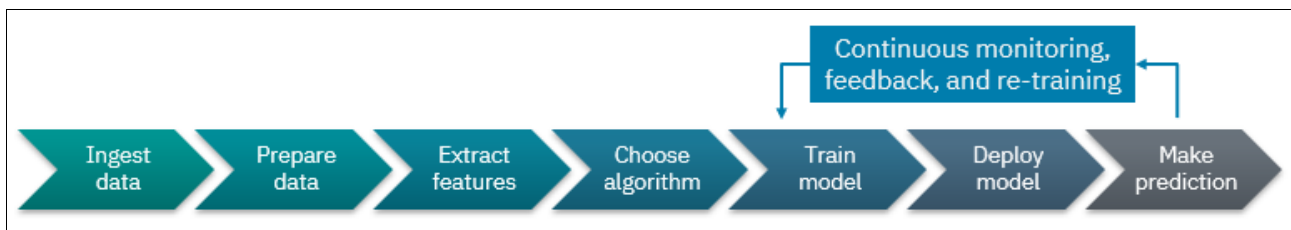


Figure 2-1 Machine learning lifecycle

Ingest data

The first step in the ML lifecycle is the data ingestion phase. This phase describes a process by which data of varying types and sources is moved and stored, or imported, into some storage medium. This ingestion can happen in real time or in batches. Those storage mediums are often data warehouses, databases, or similar products. As part of the data ingestion process, the data is cleaned and transformed, which makes it easily accessible in an analytics environment.

Prepare data

The next step is the data preparation phase, where you clean the data set by removing faulty and irrelevant data. It is also important that you deal with any missing values in the data set in this phase. Optionally, you also might want to enrich your data through the addition of data from different data sources, such as historical or personal data.

Extract features

It is common for data sets to have many features and having too many features as part of your model can lead to over-fitting of your model. *Over-fitting* occurs when the model is too complex or too specific, which results in accurate predictions on data it was trained on, but inaccurate predictions against new unseen data.

In a case of over-fitting, feature extraction is a vital part of your process. In the feature extraction phase, the goal is to reduce the number of features that is used by the model going forward. This process leads to more accurate predictions and faster training. The easiest way to achieve this goal is feature selection, where you choose a few of the features and discard the rest. Other techniques include dimensionality reduction, where you reduce the number of features by combining several features into one new feature.

Choose algorithm

After your data is ready, you move to the algorithm selection phase. Each of the approaches that we discussed can be split into methods or the types of problems they aim to solve, and those types of problems contain various types of algorithms that can be used in ML processes.

Supervised learning is split into classification and regression problems; unsupervised learning is split into clustering, association, and dimensionality reduction problems.

Often, several types of ML algorithms can be used to solve your problem. Part of the development process is to test different models and evaluate which best fits your needs.

For example, if you want to predict whether a customer might default on a loan, a Logistic Regression algorithm can be used to solve this problem. However, a Random Forest or a Neural Network also are potential solutions.

Important considerations to make when selecting an ML algorithm include evaluating your input data to know what you are working with, defining the problem that you must solve and goals for that outcome, and reviewing and testing your options when deciding on which algorithm to implement.

Recent IBM publications have shown how AI that ingests raw data can automatically perform data preparation; select the best performing ML or deep learning (DL) algorithm; and find the optimal hyper parameters to result in the “best” trained model. The IBM Research blog [AI for AI set to make it easy to create machine learning algorithms](#) demonstrates a new DL design approach that is aimed at automating the AI model creation process through optimizing the complicated and time-intensive tasks that are typically associated with AI model selection.

Ultimately, this research work led to a product enhancement of automated machine learning (AutoML) to become AutoAI.¹ It extends the automation of model building to the entire AI lifecycle. Like AutoML, AutoAI applies intelligent automation to the steps of building predictive ML models. These steps include preparing data sets for training; identifying the best type of model for the data, such as a classification or regression model; and choosing the columns of data that best support the problem that the model is solving, known as feature selection. Then, AutoAI tests several hyper parameter tuning options to reach the best result as it generates and ranks model-candidate pipelines that are based on metrics such as accuracy and precision. The best performing pipelines can be put into production to process new data and deliver predictions that are based on the model training.

¹ <https://www.ibm.com/cloud/watson-studio/autoai>

Train model

Before training your model, you must randomly split your data into training, validation, and test data. Splitting the data into these categories helps to avoid over-fitting.

During the training process, the model is fed the training data, which accounts for the largest portion of data so the model can see as many examples as possible. Then, the model is slightly adjusted to increase its accuracy. This process is repeated several times, which continuously improves the model each time, as illustrated in Figure 2-2.

Then, the validation data helps to adjust the hyper parameters and feature selection, which is sometimes referred to as *tuning the model*.

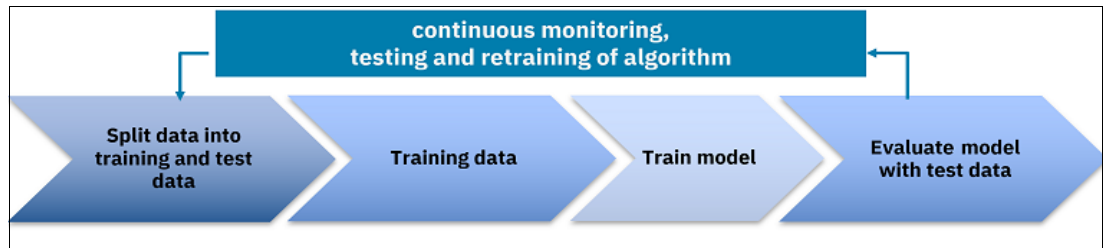


Figure 2-2 Inner loop of the ML lifecycle to continuously train and evaluate the ML model

Hyper parameters are properties that must be set before the model is trained; for example, the learning rate for neural networks. Also, hyper parameters must be optimized; that is, you must find the optimal values for the hyper parameters to improve your performance metrics.

As the algorithm is fed validation data, it learns relationships in the data and identifies which data points serve as the best predictors for your wanted outcome.

Finally, the test data is ready to be used. With the test data, you can evaluate and compare the tuned models to obtain insight into their accuracy and whether they generalize well to new data inputs. This process of selecting the best performing algorithm is call *model optimization*.

Deploy model

You want to bring in collaborators, such as software engineers or business analysts, to validate the quality of the model. After you develop, build, train, and select the most optimal model, you want to make it available to applications and users, a process that also is known as *deployment*.

Two common types of deployment are available:

- ▶ Online deployment involves real-time requests and responses where the model is started by using a Representational State Transfer (REST) application programming interface (API).
- ▶ Batch deployment involves reading and writing from a static data source and also can be started by using a REST API.

When selecting a deployment method, it is important to consider how often you must make predictions and whether any latency requirements must be met. You also must consider the number of applications that need access to the model, and whether will they score a single input or batch inputs.

If you need several applications to access the model, consider a single online deployment of the model and make it accessible to all applications by way of an API. For example, deploy your model on a web server and provide a REST API to be called from applications.

However, you might want the model to score numerous data in batches on a weekly cadence. In this case, it is likely better to implement a batch deployment locally, and potentially as a stand-alone application.

Make prediction

This final stage of the ML lifecycle is about the use of your model in production and adding value to your business. To make a prediction, the steps that are in Figure 2-3 must be repeated, similar to the repetition that is performed in model training.

Note: In this context, making a prediction is synonymous with obtaining any outcome, which can be a score or a classification instead of a prediction.

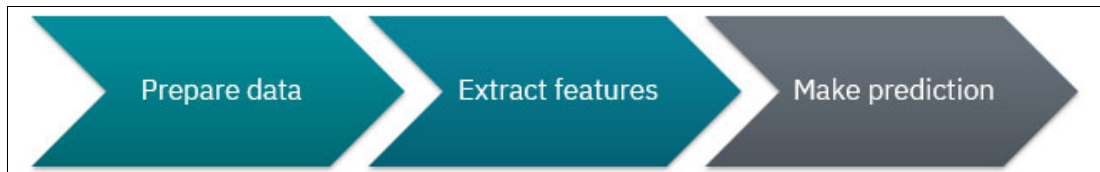


Figure 2-3 Making a prediction

The first step is to enrich and prepare the model input data that is in your application. Then, the features must be extracted. These steps must be conducted in the same manner as they were during model training so that your model can correctly interpret the inputs.

It is important to ensure that the data that your model is importing is relevant to the problem that the model is trying to solve and whether new, more relevant data sets are available to help improve the accuracy of predictions. Then, you can begin making predictions based on new input data, which looks different depending on your deployment scenario.

Continuous monitoring, feedback, and retraining

While your model is deployed in production, you must monitor and manage it as with any other software. You look for errors and track latency, performance, accuracy drift, bias, and business key performance indicators (KPIs).

You also must track the accuracy of your model because the model accuracy can deteriorate over time as your test data shifts from the original data that was used to train the model (referred to as *model drift*). All the information that you collect as a part of monitoring and managing the model serves as feedback to the reliability and explainability of the model. This feedback enables fine-tuning of the model through model retraining.

You can retrain a model through a direct data retraining approach that uses more recent and relevant data, or through a full lifecycle retraining approach, which retrains the model through all stages of the ML lifecycle. Continuous monitoring, feedback, and retraining as described in Figure 2-2 on page 22 is vital to maintaining trust in the model through transparent prediction making and mitigating bias. The inability to understand how and why a model makes a prediction and the concern around bias in AI can expose your business to significant reputational and financial risk.

2.2 Integration and deployment best practices

The methodology of developing models is the foundation and core of an AI project; however, having trusted and accurate models is not the final success of an AI implementation.

The goal of building ML models is to solve a business problem. This process is achieved only when the models are deployed to production and actively used by applications for business decisions. When we say *deployment*, we are referring to the process of configuring the model to integrate with other applications or user access.

Because difficulties with the deployment process are a major inhibitor of enterprise AI adoption, it is important to understand the details regarding these key challenges and best practice solutions for integration into the production environment.

2.2.1 Model interoperability

One key challenge of ML model integration and deployment is the infrastructure difference between the model training environment and the production environment, especially when the goal of the project is to optimize business decisions in applications by using AI.

Today, many different ML and DL frameworks are available for data scientists to use for model development. These frameworks include scikit-learn, TensorFlow, PyTorch, Caffe, and programming languages such as Python, R, and Java.

In a production environment (especially for traditional industries, such as banking, insurance, health care, and government), we mainly are referring to online transaction processing (OLTP) and online analytical processing (OLAP) applications. These OLTP and OLAP applications are written in different programming languages such as Java, COBOL, C, and run on different platforms, such as Linux, cloud, and IBM zSystems.

It is not practical, even impossible, to rebuild production infrastructure to use different types of ML models. Instead, it is a best practice to convert the framework and language-specific models into some exchangeable format. Doing so ensures that models are interoperable among the training environment and different production deployment targets.

The models can work with these other environments and targets. As of this writing, the most widely used exchangeable formats for model interoperability are Predictive Model Markup Language (PMML) and Open Neural Network Exchange (ONNX).

Predictive Model Markup Language

PMML is an XML-based file format for describing ML models and pipelines. Many ML frameworks and tools, such as scikit-learn, R Studio, IBM SPSS Modeler, and SAS support converting and persisting models in PMML format.

Open Neural Network Exchange

ONNX is a standard format that also is used for ML models, and it is especially versatile for representing DL models. It defines a set of common operators and a common file format that enables the portability of DL models across frameworks and platforms. Because of this feature, ONNX is a key technology in supporting the IBM zSystems AI strategy to enable clients to build and train models anywhere, and perform a simple deployment on IBM zSystems with seamless optimization (see Figure 2-4 on page 25).

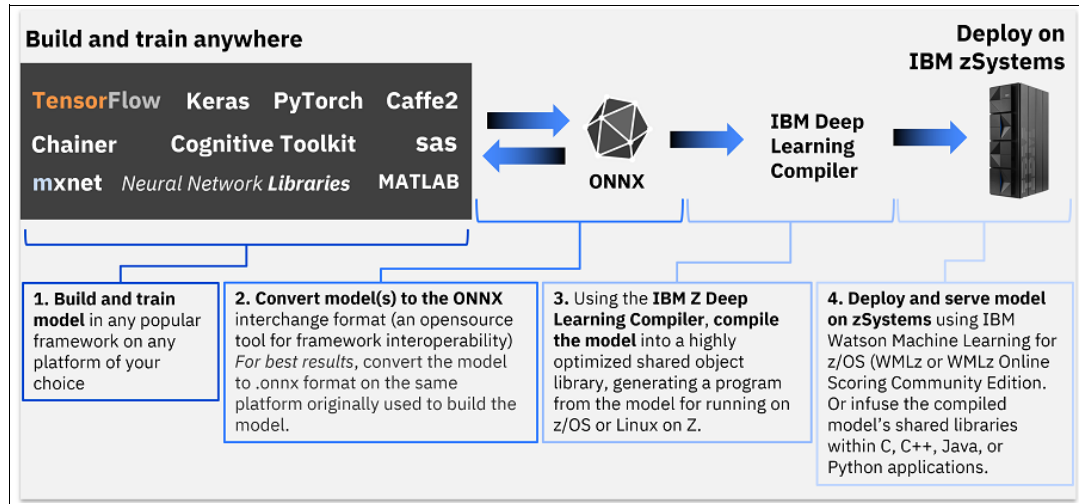


Figure 2-4 Deep learning ecosystem on IBM zSystems

Today, most available DL frameworks support the conversion of models that were trained with a specific framework to the standard ONNX format by using the command-line interface (CLI) or APIs. For example, *tf2onnx* is one of the most popular open-source projects that can convert models that are trained by TensorFlow and Keras to ONNX, which makes them portable to IBM zSystems.

In addition to TensorFlow, PyTorch is a popular DL framework that supports the direct export of models to the ONNX format.

With ONNX, DL models that are created by using different frameworks can be deployed to a unified environment for inferencing so that model development and model deployment processes are decoupled. It also provides flexibility to data scientists to use their skills and choose the model development framework, while also easing the model deployment to production process.

For more information about exporting or converting a model to ONNX format and to access the *tf2onnx* converter, see [TensorFlow-ONNX GitHub](#).

2.2.2 Piloting your first project

Often, data scientists analyze a problem and decide on a modeling technique that is based on the characteristics of the problem. In these circumstances, you might be tasked with deploying the model, as described in 3.2.2, “Credit risk model deployment and evaluation on IBM zSystems” on page 55.

If you are exploring the usage of ML to solve a problem, you must establish where the opportunity for growth exists within your enterprise. As a best practice, start with a small project that is directly connected to a business outcome.

Another important consideration to make when looking at opportunities for growth and deciding which problem might benefit most from your ML project is to have a good idea of the data that is readily available and the data that might be made available to enable the success of the project. Understanding this situation enables you to better realize how that data can help solve your problem and the nature of the problem. Start with writing out the business problem, the type of data that is to be used, and the purpose of your ML project.

As a best practice, use data that you have on hand or that can be collected without a significant investment of time. The data also should be data that requires little cleaning and preparation.

The feature extraction and data manipulation must be handled manually until your project proves successful and you move on to more robust ML projects. As a best practice, start your ML effort with a supervised learning approach because that approach is considered easier to understand when compared to other approaches. Also, the value that the predictions bring to your enterprise can be more easily assessed with a supervised learning approach. As described in “Choose algorithm” on page 21, use tools that are available to help decide the algorithmic approach that is best suited for your specific use case.

Depending on the amount of data that you plan to use for training the model, the training can be performed by using a single workstation or a cloud computing service. Model performance depends on the data scientists performing the training, their skill set, and their domain knowledge of the business challenge.

Various open source frameworks such as scikit-learn, TensorFlow, PyTorch, and Keras can be used to develop and train your model, which can then be seamlessly deployed to IBM zSystems. Several open-source integrated development environments (IDEs) also are available to provide an easy-to-use interface for training a model, such as Jupyter Notebooks, RStudio, and Google Colab.

To enable collaboration in model training, data scientists use open-source version control systems, such as Git, or a Git-based version control system for ML. Conversely, you can use products such as IBM Cloud Pak for Data (CP4D), Watson Studio, and IBM SPSS Modeler to enable AutoAI functions, such as hyper parameter optimization (HPO) and feature selection, open-source integration enablement, and a collaborative environment to support the different phases of your ML lifecycle.

As a best practice, serve your trained model as a service by using a REST API so that you are not restricted in accessing the model, and can easily leverage the model from anywhere.

This type of pilot project likely serves as marketing material inside the enterprise, and it can be used to convince and demonstrate to decision makers how ML can be applied to your business problems and result in significant improvements. The project also can serve as a stepping stone to solving a much larger problem within the enterprise by building on the project and by using everything that was learned from it. You might even discover some valuable insights while developing your pilot project that might not be noticed otherwise.

Pilot projects equip the enterprise with a clear, unbiased understanding of the business in a unique way, which helps evaluate assumptions that are made by the business about trends and customers. It also shows the benefits of applying ML to solve a business problem.

2.2.3 IBM Watson Machine Learning for z/OS overview and integration points

IBM Watson Machine Learning for z/OS (WMLz) is an enterprise-grade ML platform that enables embedding ML and DL models in transactional applications for real-time insights. WMLz is a production-ready ML platform that delivers several essential enterprise-grade ML features, such as model versioning, auditing, and monitoring. Additionally, WMLz is designed for high availability (HA), strong security, high performance, and low inference latency.

As the best unified ML platform on z/OS, WMLz automates the entire flow of ONNX model import, compilation, deployment, and scoring by using the IBM zSystems Deep Learning Compiler (zDLC) transparently. The WMLz ONNX engine supports Version 13 of the ONNX operation set (opset). When importing DL models with WMLz, the model automatically compiles and optimizes for the IBM zSystems Integrated Accelerator for AI, if it is present. PMML models can be configured with a similar flow to that of ONNX models.

By using WMLz, you can easily integrate PMML, ONNX, and CP4D models that are trained on platforms other than IBM zSystems. The built-in online scoring service of WMLz includes the zDLC capability and features an accessible interface that is used to easily upload and deploy models for inferencing. It also makes available a REST API for each deployed model, which can be called from z/OS applications.

Any applications that are running on IBM zSystems can use the unified scoring REST API for real-time model inferencing after obtaining the scoring endpoint from the WMLz GUI. Also, WMLz allows you to configure the online scoring service in an IBM Customer Information Control System (IBM CICS) region by using data gravity as a result of the colocation of your transactions and scoring service. Then, you can analyze that data where it originates to drive better business decisions.

Supported data sources in WMLz are determined by the type of data access method that are you are using: Java Database Connectivity (JDBC) or Mainframe Data Service (MDS). By using the JDBC data access method, WMLz supports access to Db2 for z/OS data sources in Scala and Python. By using the MDS data access method, WMLz supports access to Db2 for z/OS, IMS, SMF, and VSAM data sources in Scala and Python.

When you take advantage of the data gravity attribute of your approach, you close the gap between where the data is generated and where insights and decisions are made. This closure results in lower costs, simpler infrastructure, security improvements with less data movement, and better data governance.

WMLz architecture and components

WMLz is composed of two elements: a required base component (WMLz base) and an optional data science integrated development environment (IDE) (WMLz IDE). An architectural diagram of WMLz is shown in Figure 2-5, which shows the interaction between the two elements and their respective components. The WMLz base is deployed on IBM z/OS and consists of a user interface, online and batch scoring services, model and deployment management services, and IBM Open Data Analytics for z/OS (IzODA).

As a bundled component, IzODA serves as the data processing cluster for WMLz, and it provides advanced data analytics capabilities through z/OS Spark (Spark), z/OS Anaconda (Anaconda), and Mainframe Data Service (MDS). Spark provides the analytics engine for large-scale data processing; Anaconda provides various Python packages for model training; and MDS provides the ability to access z/OS data sources and connect them to the data processing engines.

WMLz IDE runs on Red Hat OpenShift Container Platform and can be installed on an s390x or x86 server. With the WMLz IDE, data scientists can easily work with data originating on IBM zSystems and use several ML libraries to create and train models. Using Scala or Python within the WMLz IDE, MDS can be used to access Db2 for z/OS, IMS, System Management Facility (SMF), and VSAM data sets. A data scientist can create collaborative project environments, build Jupyter Notebooks, and create SPSS Modeler flows. Models that are created in the WMLz IDE can be easily imported into WMLz base for management and deployment.

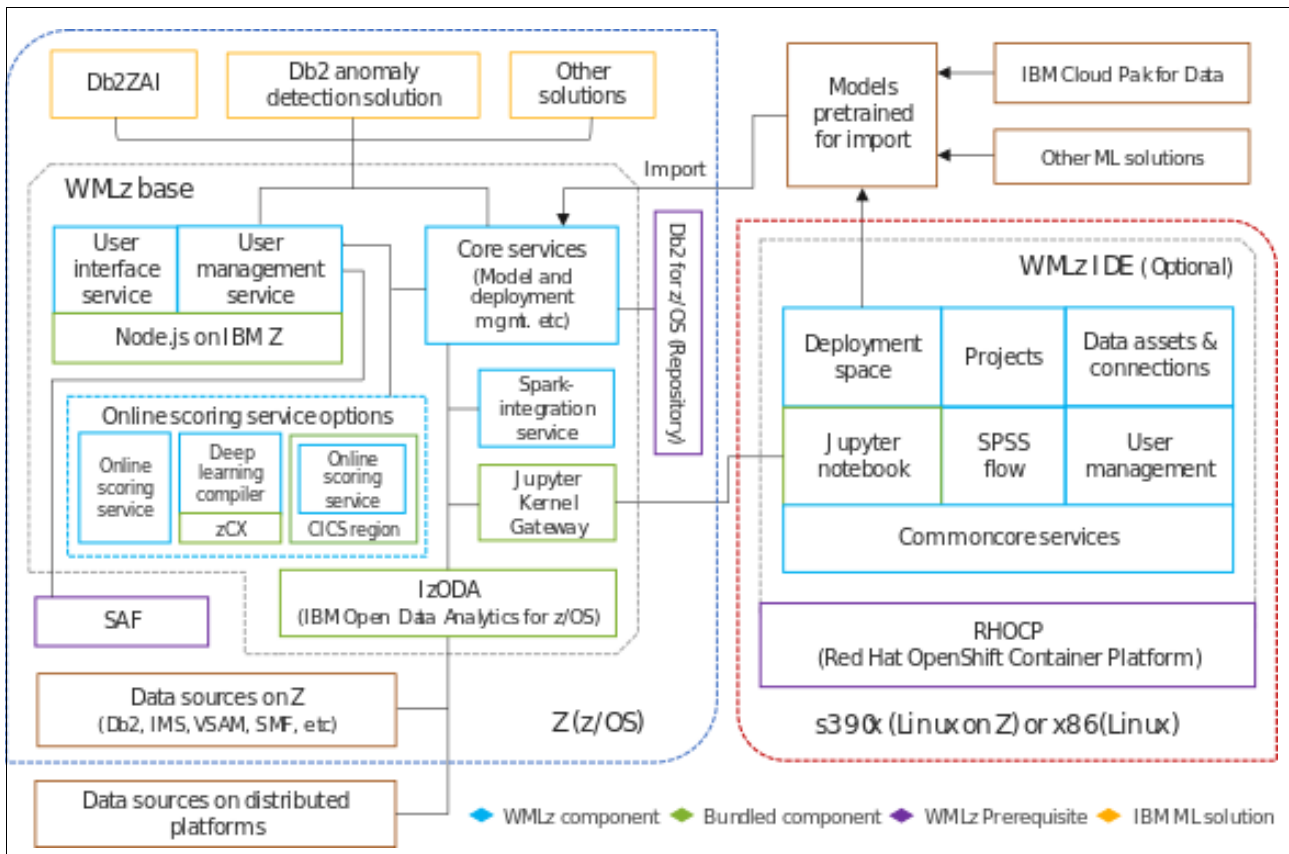


Figure 2-5 IBM WMLz architecture diagram

At the time of writing, WMLz IDE supports creating and deploying MLeap, SparkML, scikit-learn, XGBoost, ARIMA, and Seasonal ARIMA models in the integrated Jupyter Notebook editor. In addition, there are several model types that can be created by the integrated SPSS flow editor.

Note: At the time the writing, IBM z/OS Container Extensions (zCX) is a required component for compiling the ONNX model into executable files for scoring. Scoring itself does not require zCX because the scoring that uses the compiled executable files is natively on z/OS. The IBM WMLz development team plans to release a product temporary fix (PTF) enhancement that lifts the requirement of zCX so that it instead supports running the ONNX compilation procedure on Linux on IBM zSystems server. For more information about IBM WMLz 2.4 PTFs, see [IBM WMLz Customer Support](#).

For more information and nuances about the supported model types, see [Supported algorithms, data sources, data types, and model types](#).

WMLz deployment features

With WMLz, several enterprise-grade deployment features are provided. Models can be deployed to highly available (HA) and performant online scoring servers in z/OS or within a CICS region. After a model is deployed, you can monitor the amount of inferencing requests coming to the deployment, and the average inferencing latency.

A downloadable API specification is provided that describes how your applications use the API. This API specification is in the format of a Swagger YAML file that describes the operations that the API supports, the needed parameters, usage of authorization tokens, the input format that is expected, and the output format that is returned. The Swagger file's machine-readable format is advantageous because it can be used to greatly simplify the embedding of API calls within z/OS applications by using a service such as IBM Integration Bus or ZIBM z/OS Connect EE.

WMLz also supports model deployment monitoring through IBM Watson OpenScale in CP4D. With IBM Watson OpenScale, an enterprise can analyze AI model deployments to transparently understand and explain why an AI model is making certain predictions. Additionally, WMLz supports deployment monitoring for bias and accuracy degradation, which are critical components for trustworthy AI, by evaluating and re-evaluating a model to combat accuracy drift. SparkML and PMML model deployments can be scheduled for periodic re-evaluation to continuously ensure adequate accuracy. The results of the evaluations can be seen on the WMLz dashboard, and SparkML models can be automatically retrained if the result of the evaluation is not satisfactory. For more information, see 2.2.5, "Continuous integration and continuous delivery in ML" on page 33.

During the ONNX model deployment process, WMLz offers the option of *micro-batching*, where the WMLz ONNX scoring engine queues a configurable quantity of inferencing requests before sending them to the IBM z16 Integrated Accelerator for AI. By processing multiple inferencing requests at once, the IBM zSystems Integrated Accelerator for AI maximizes throughput and reduces the response latency compared to processing inferencing requests one at a time. In addition to the customizable micro-batching quantity, a *maximum latency time* can also be defined for the deployment, which means that the scoring server waits only a certain amount of time before passing the accumulated inferencing requests to the Integrated Accelerator for AI.

Note: For a model to be eligible for micro-batching, it must have one input dimension that is dynamic and one output dimension that is dynamic.

For more information about implementing micro-batching, see “Reshaping an ONNX model to take advantage of WMLz micro-batching” on page 69.

IBM zSystems Deep Learning Compiler and WMLz

The IBM zSystems platform is suitable for model interoperability through support for open standards. By using the solutions that are available on the platform, you can seamlessly integrate traditional ML and DL models that are trained on distributed platforms to IBM zSystems for inferencing.

When deploying an ML model, use technology that enables you to create an inference solution that is suitable for IBM zSystems workloads. The zDLC, sometimes referred to as the *ONNX model compiler*, is a key component in achieving the goal of a low latency AI inference solution that meets service-level agreement (SLA) requirements while running alongside these business-critical workloads.

As key project organizers, IBM Research leads the zDLC development effort by building on the ONNX-Multi-Level Intermediate Representation (MLIR) open-source project. The compiler optimizes ONNX model inferencing, which is Multi-Level Intermediate Representation (MLIR)-based and used to transform your ONNX model to a highly optimized shared object library that can target the latest IBM zSystems hardware optimizations. This highly optimized shared object library represents an optimized, lightweight, and minimal program that can be started for inferencing that is based on the input ONNX model when output by the compiler.

Recent enhancements in the zDLC design enable models that are compiled by using the zDLC to quickly and easily take advantage of the IBM Telum processor's Integrated Accelerator for AI inferencing capabilities, in addition to other benefits, such as increased throughput and reduced latency. The compiled models take advantage of IBM zSystems technologies, including single instruction, multiple data (SIMD) on IBM z13⁰ and later, and the Integrated Accelerator for AI, which is available on IBM z16, without requiring changes to the original model. The Telum processor enables real-time embedded AI directly in transactional workloads, which further accelerates inference performance for high-volume workloads at scale for the next generation of the IBM zSystems platform.

Also, the compiler is an integrated capability for WMLz to improve deployment efficiency. For more information about the ONNX-MLIR open source project, see [ONNX-MLIR GitHub](#).

Comparing WMLz Online Scoring Community Edition and WMLz 2.4

The WMLz Online Scoring Community Edition (WMLz OSCE) is a special no-charge, feature-limited version of WMLz that is intended for simple testing of WMLz real-time scoring of pretrained ONNX models. WMLz OSCE is packaged as an s390x Docker container image that is easily deployed in an zCX. WMLz OSCE can be used for rapid use case evaluation of embedding DL models in transactional z/OS applications. The scoring endpoint is deployed within zCX, and it can be accessed by z/OS applications through a RESTful API call.

WMLz OSCE is not intended for enterprise production use, and it has a much more limited feature set than WMLz.

For more information about the features that are included in WMLz and WMLz OSCE, see [What's new in WML for z/OS 2.4.0?](#)

2.2.4 Batch versus online inference

As with model deployment approaches, model inferencing also uses batch and online approaches. To decide whether batch inference or online inference is best for your models, you first must have a solid understanding of your use case. The answer can vary by case, depending on the business problem that is being addressed.

The inference approach that you select is important because it leads to varying infrastructure requirements. The differences between batch inference and online inference are discussed next.

Batch inference

Batch inference, or offline inference, is the process of generating predictions for several records at once by using batch jobs. Batch jobs for inferencing are normally scheduled to run regularly (for example, hourly, daily, or weekly) based on the business requirements.

Because of its recurring nature, batch inference cannot be used for real-time predictions. Typically, the prediction results that are generated by batch inference jobs are written to and stored in database tables so that they be further analyzed by data analysts or processed by other applications.

As an offline process, batch inference can run from minutes to hours. If it can fit into the batch window, which most latency requirements do, latency is not a significant concern.

Batch inference also tends to be CPU and memory intensive because of the large volumes of records being processed simultaneously. To mitigate this issue, you can use big data technologies, such as Spark, to optimize the performance and lower the cost when building up your infrastructure for batch inference.

Online inference

Conversely, we have *online inference*, which is the process of generating predictions on demand, in real time. Typically, online inference is embedded in online transactions as a synchronized process that processes one or several records for prediction at a time.

Online inference unlocks an entire new space for traditional applications to benefit from ML with real-time insights as needed. Compared with batch inference, online inference tends to be more complex because the inference engine must meet the response time, scalability, and HA requirements and enforce transaction SLAs.

Batch versus online inference process in WMLz

When planning for the production integration of your AI models, you must clearly define which inference strategy you plan to implement for the models.

Generally, online inference is required whenever predictions are needed for on demand, real-time insights. For example, if your goal is to embed AI in your instant payment transactions to enable fraud detection in real time, you must use online inference because you need the prediction results that are returned in several milliseconds to identify and prevent fraudulent activity as soon as possible.

However, batch inference should be used in a case where latency is not critical. For example, predicting customer churn analysis, where it is sufficient to get the prediction results daily, or even weekly, for your customer retention actions.

After you decide on the inference strategy, the next step is to set up the infrastructure to support your wanted goal. You can use WMLz to fulfill your needs for batch inference and online inference (see Figure 2-6).

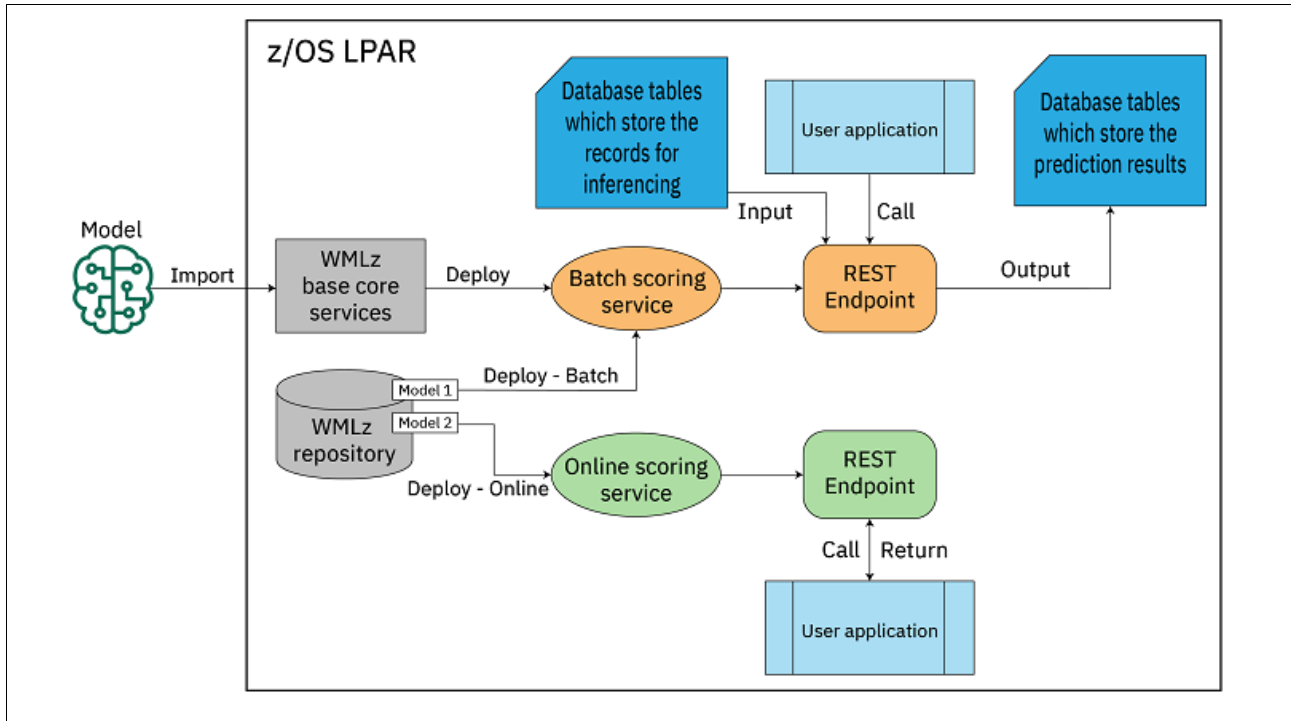


Figure 2-6 WMLz batch inference versus online inference process

The process begins when you import your models into your WMLz base component, or if it was previously imported into your WMLz model repository. Then, after you deploy a model from either starting point, you choose your deployment type (online or batch).

The WMLz base component includes batch and online scoring services, which expose unique REST endpoints that are accessed through REST APIs for each deployed model. Batch scoring then takes the group of records from database tables as input, and writes the prediction results for the group back to the database tables for storing and access by the application.

The WMLz batch scoring service uses Spark for z/OS for high performance, zIIP eligible, in-memory data processing. WMLz online scoring service processes one or several new records as input in real time, and then, returns the prediction for direct access by the application and provides optimized performance for different types of models.

WMLz online scoring services can support low latency inferencing, even in the single digit milliseconds for some models. Factors, such as model complexity, greatly affect the latency that can be achieved.

WMLz also supports HA scoring clusters for online inference and batch inference to reduce the downtime of your applications that are running on z/OS.

Figure 2-7 shows an example architecture that includes an online scoring cluster that is deployed on a SYSPLEX with two logical partitions (LPARs).

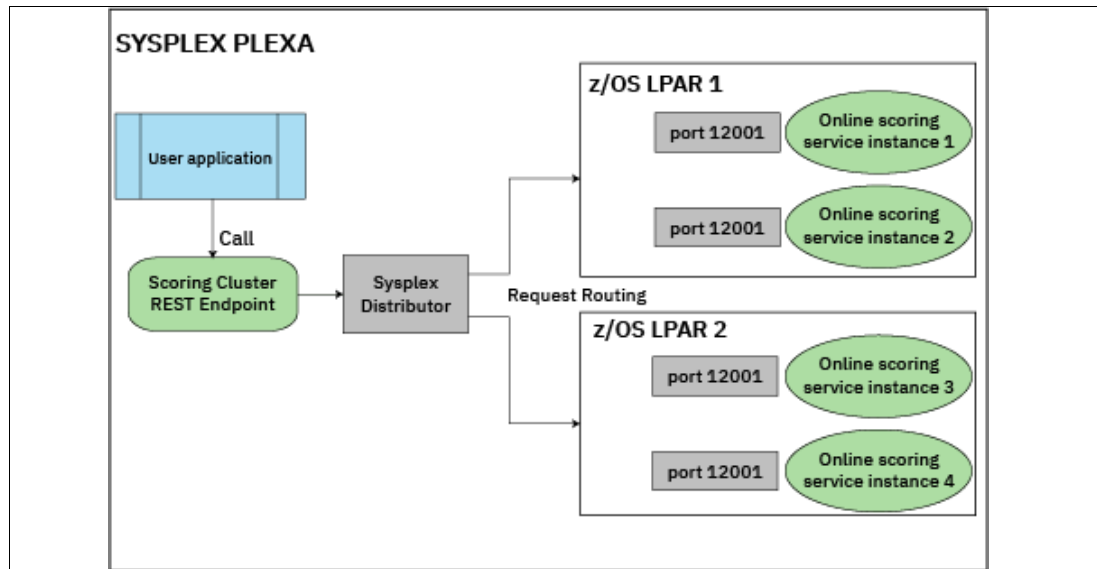


Figure 2-7 WMLz online scoring cluster example

The scoring cluster has four scoring service instances, which are separated by two instances on each LPAR. The application calls the scoring cluster through one REST endpoint, which is then routed to available scoring instances by using Sysplex Distributor capabilities.

The Sysplex Distributor is used to provide TCP/IP load-balancing between multiple LPARs to optimally distribute requests to achieve and optimize workload balancing. In addition to HA, scoring clusters also can help enable scalability to support high transaction throughputs.

2.2.5 Continuous integration and continuous delivery in ML

Because models are trained by using historical data, the model is likely to experience model drift where the accuracy of a deployed model degrades over time as new data flows are introduced to the algorithm. It is essential to the integrity of your AI systems that you continuously monitor, retrain, test, and integrate into the production environment every model that was deployed.

Integrating an infrastructure and automated process to manage the essential iterative process of maintaining your models lifecycle can be critical to the success of your AI implementation over time. This supporting infrastructure is the core of continuous integration and continuous delivery (CI/CD) in ML.

Data access

The first part of your pipeline is having the infrastructure to support access of various volumes and types of data from multiple different sources. Data growth continues to occur at an exponential rate with a growing number of data storage mediums, which results in an increase in the complexity of data, and data movement security risks, expenses, and efforts.

Data Virtualization is a recommended approach to data access over traditional data movement approaches to avoid outdated insights and security risks. It connects multiple data sources into a single self-balancing collection of data sources or databases, which enables independently designed data structures to be used together with minimal added processing costs.

IBM offers the IBM Data Virtualization Manager for z/OS (DVM) for virtual, integrated views of data that is the IBM zSystems platform. This tool eliminates the need to move, replicate, or transform data instead enabling data access by using industry-standard APIs (when combined with IBM z/OS Connect).

In turn, you can enrich your insights with live, transactional data at speed, and benefit from DVM natively running on IBM zSystems zIIP processors. DVM enables access to many varying data sources, such as VSAM, IMS, ADABAS, IBM MQ, and non IBM zSystems data.

DVM also provides access to almost any application by using industry-standard APIs, including JSON, Structured Query Language (SQL), SOAP, and REST, by using IBM zSystems hardware benefits. This access to native IBM zSystems data sources in real time helps simplify your ML model development from IBM zSystems data and reduces the costs that are associated with moving data to other platforms.

For more information about DVM, see *IBM Data Virtualization Manager for z/OS*, SG24-8514.

Building on the capabilities of Data Virtualization is CP4D 4.5.x. CP4D provides a connection to different data sources, such as IBM Db2, Oracle, Hive, Microsoft SQL Server, Teradata, and Hadoop.

The Data Virtualization service in CP4D virtualizes the data by taking data with similar sources and joining them into unified virtual tables through the extract, transform, and load process. The service then groups the tables by schema, which are then ready for data governance rules and project associations to be applied before being ready for use by projects and applications. This process allows querying across multiple data sources in the same way you query against a single data source, without compromising query performance by using parallel processing.

When you incorporate CP4D into your CI/CD pipeline, you are equipped with a central location for access control and data governance. You can connect and query multiple data sources and use the virtualized data, which breaks down data silos and simplifies data analytics throughout the enterprise.

For more information about data virtualization tools and solutions on IBM zSystems, see [IBM Data virtualization tools and solutions](#).

Developing and deploying ML assets in a CI/CD workflow across z/OS by using WMLz

IBM recognizes the critical requirement of allowing businesses to keep their sensitive, mission-critical data where they want it and providing solutions that bring cognitive capabilities to the data. WMLz is the solution that enables you to use ML capabilities on the IBM zSystems platform to extract actionable insights from sensitive, mission-critical data while keeping the data in the security-rich IBM zSystems environment and leveraging existing IBM zSystems capabilities. It provides an enterprise-grade ML solution on the IBM zSystems platform for running ML with data in place, with HA, and developer-friendly APIs for applications on the IBM zSystems and Red Hat OpenShift Container platforms.

HA and real-time predictions with minimum impact to transaction processing are the key requirements for enterprises looking to leverage ML on the IBM zSystems platform. To enable these predictions, WMLz takes this requirement to heart in the architecture of the scoring services that run natively on z/OS, bringing state-of-the-art ML technology behind the firewall on IBM zSystems. WMLz also provides powerful advanced features to assist the integration of your model lifecycle management into your enterprise's current CI/CD pipeline as part of your DevOps process, which significantly reduces and simplifies development and deployment tasks.

During model development, you can use IBM Cognitive Assistant for Data Scientists (CADS), which automates the selection of the best performing algorithm from your algorithm candidates for the model, which are accessible from WMLz by using the supported Scala APIs and CADS APIs. In addition to CADS, you also can use the HPO estimator, which is accessed by using the CADS API and automatically selects the hyper parameter that helps produce the best-performing model for the algorithm.

The ability that is enabled by WMLz to use advanced proprietary cognitive technologies from IBM Research while seamlessly integrating with the most commonly used open-source packages with which data scientists are familiar makes it possible to condense a weeks-long process into days or hours. For more information about the technical details regarding the WMLz CADS APIs and the HPO estimator, including examples, see [IBM CADS \(Cognitive Assistant for Data Scientists\) API](#).

You can even automate feature recommendations by using the capabilities of WMLz to select relevant features based on correlation analysis. Also, WMLz provides APIs that can be called to store, deploy, update, and delete models in the repository, which provides access to other tools and processes in the pipeline and enables automated deployments to test and production environments (see Figure 2-8).

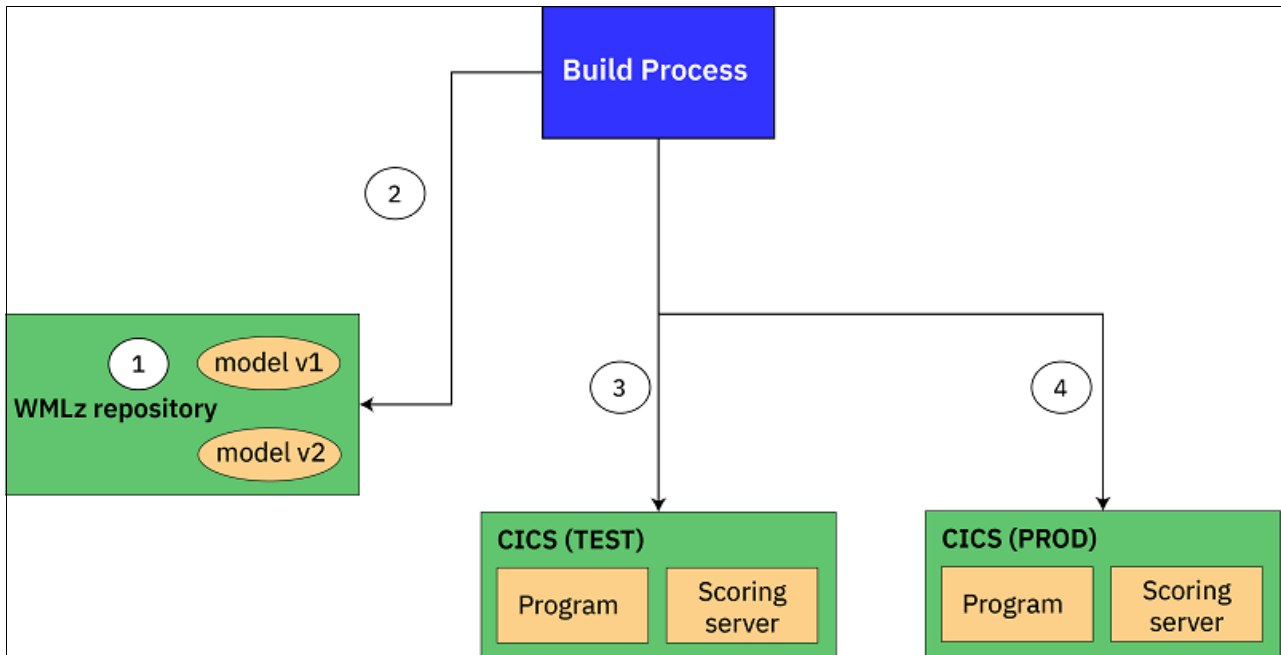


Figure 2-8 Model lifecycle management by using WMLz

When integrating ML models into the production system, data scientists can choose to enable the automated retraining option. With this feature enabled, WMLz continuously monitors the performance of the model. It automatically triggers retraining after it detects the degradation of the accuracy of the model reached a specified threshold. The auto-retrain then generates a new version of the model and stores it in the WMLz repository.

CP4D can be used with WMLz to enable the access and management of all enterprise data on and off the IBM zSystems platform. Another benefit of the usage of the tools together is the many colocation scenarios that you can implement, which influences data gravity by reducing data movement. You can use DVM for z/OS to provision IBM zSystems data to CP4D for Data Virtualization. Together, they offer a powerful combination of capabilities to manage and use your data and AI.

The CI/CD pipeline build process can call the API that is provided by your WMLz service to detect whether a new version was generated for the deployed model. After it detects a new model version is available, the build process uses the WMLz API to deploy the new version to the scoring server that is running in the testing environment. After testing of the new model version is complete, the build process calls the WMLz service to integrate and deploy the new version to the production system. This entire process can be automated, which requires no human intervention.

The model lifecycle is much larger than what this chapter covers, so your CI/CD solution for ML is likely to be much more robust than what was described here. For more information about the tools and capabilities to enable the usage of ML in your CI/CD pipeline at each stage of your model lifecycle, see [AI Model Lifecycle Management: Overview](#).



Real-time, in-transaction scoring use case scenarios

Real-time, in-transaction analytics provides companies the ability to truly tap into their rich vein of historical system of record information. It enables mining to automatically discover insights and generate predictive models with scoring capabilities to take advantage of all the data that they are capturing. Therefore, instead of looking into the past for generating reports, businesses can predict what might occur in the future based on analysis of their data.

Having this ability makes it possible to achieve many benefits, such as personalizing every client interaction, reducing risks, detecting fraud, cross and up selling, customer categorization, inventory optimization, and many others that are meant to increase your revenue and disrupt competition.

This chapter describes the business value of real-time, in-transaction artificial intelligence (AI) scoring for various industries and how IBM zSystems hardware and software enable an enterprise to inference all its business transactions. We describe and demonstrate the end-to-end implementation of two use cases that are focused on real-time, in-transaction scoring. In the first use case scenario, we explore integrating a credit risk AI model with an IBM Customer Information Control System (IBM CICS) application. In the second use case scenario, we demonstrate training and deploying a deep learning (DL) credit card fraud model into IBM Watson Machine Learning for z/OS (WMLz).

This chapter includes the following topics:

- ▶ 3.1, “Business value of in-transaction scoring” on page 38
- ▶ 3.2, “Credit risk assessment use case scenario” on page 48
- ▶ 3.3, “Credit card payments fraud detection use case scenario” on page 60
- ▶ 3.4, “Summary” on page 78

3.1 Business value of in-transaction scoring

The value of real-time, in-transaction scoring is rooted in its ability to use the most recent and relevant available data to uncover valuable business insights. This process leads to trusted, actionable results and the development of more accurate machine learning (ML) models to make optimal decisions that drive business value.

More simple solutions generally include the use of post-transaction analytics with aged data, which results in suboptimal decision making. It also can lead to missed opportunities and decreased customer satisfaction and retention.

Real-time transactional scoring can arm your enterprise with actionable insights on every transaction, increase revenue, reduce risk, and improve trust with access to the most current data for the best analytical outcome, all while maintaining service-level agreements (SLAs). Real-time in-transaction scoring delivers better, more profitable decisions that use the latest data at the point of customer impact and in turn enables the following benefits:

- ▶ Engage in more informed customer interactions, which improve customer service, and increases revenue per customer ratio. It also leads to more personalized recommendations that increase cross and up selling opportunities, improves effectiveness, and heightens customer retention.
- ▶ Improves accurate fraud identification and prevention efforts with the latest data, which in turn reduces risk and vulnerability to new approaches to fraud, and cost of recovery actions.
- ▶ Improvements in the speed and accuracy of scoring to drive better business insights and profitable decisions by using new and relevant data directly within the online transaction processing (OLTP) application.
- ▶ Deliver the performance that is required to meet and exceed the SLAs of OLTP applications.
- ▶ Improve resource usage, which minimizes demand on the network, hardware, software, and resources.
- ▶ Discover and drive new business opportunities through better business insights and enable quick responses to market changes before it impacts the bottom line.
- ▶ A single infrastructure for reduced complexity and redundancy of hardware software, and administration resources.
- ▶ AI systems are more reliable by leveraging the same high qualities of service as the OLTP systems on which they are deployed.

3.1.1 Industry-specific scenarios that use in-transaction scoring

The following industry-specific use cases use ML and real-time in-transaction scoring to obtain valuable business insights and benefits:

- ▶ Health care: Aided diagnosis, and disease prevention
- ▶ Finance: Payment and credit card fraud prevention, financial trade optimization, streamline account setup and customer acquisition, automate compliance, loan approvals, and reporting
- ▶ Insurance: Faster claim validation, underwriting processing, policy issuance, and fraud prevention
- ▶ Retail: Marketing personalization, and improved customer service

- ▶ Security: Security screening optimization, and improved cybersecurity
- ▶ Media and Entertainment: Ad targeting, and audience prediction
- ▶ Utilities: Usage pattern analysis, and identify efficiency opportunities
- ▶ Telecom: Customer churn and retention, and network performance
- ▶ Transportation: Self-driving cars, and traffic congestion prediction
- ▶ Government: Automate claims and benefits processing, and verify property appraisals

An in-transaction scoring solution that is embedded with ML can be implemented for the following various use cases within an enterprise to solve business problems:

- ▶ Fraud detection

An implementation example of an anti-fraud use case is embedding point-of-sale transaction processing with fraud detection, which means predicting whether a customer transaction (for example a credit card charge) is fraudulent and should be blocked or flagged by the financial institution. Then, based on the risk level, minimize or eradicate in real time the effect and cost of fraudulent transactions for the institution and customers.

- ▶ Anti-money laundering (AML)

An example includes an implementation of ML and fuzzy matching to augment in-transaction monitoring to improve and scale AML compliance efforts. This implementation enables better prioritizing of risks and reduces false positives of suspicious transaction alerts, which in turn allows analysts to focus on suspicious behaviors and quickly resolve low-risk alerts. It also significantly speeds up AML investigations.

- ▶ Risk aggregations

An example includes streamlining and automating risk reporting processes, which enable reports to be generated with data from multiple perspectives in an integrated view for a simpler, more comprehensive understanding. This ability helps satisfy risk reporting requirements with less effort and improve performance issues with accurate, holistic risk assessments across the enterprise.

- ▶ Credit risk assessment

This type of assessment automates the credit risk assessment process of customers and achieves a better understanding of the results through decision-making transparency. This process improves trust between the financial institution and the customer, increases loan application processing speed, and obtains more accurate credit risk assessment results before any lending is approved.

- ▶ Third-party risk management

When engaging in business with third parties, such as vendors and subcontractors, a scalable process can be achieved for managing the risk and compliance that is associated with such third parties. This process can lead to a clearer understanding of how third parties relate to key business processes. It also enables the enterprise to effectively and quickly identify and resolve third-party risks or regulatory compliance issues through activity monitoring across the enterprise.

- ▶ Treasury or currency risk management

By using pre-trade risk analysis and continuous risk modeling trade scenario implications on portfolios can be predicted, including the consideration of the cost of risk capital, while understanding risk exposure in real time. This ability enables the enterprise to mitigate the rising costs of inefficient margin calculations and present the most accurate capital-ratio impact predictions to customers before trades are completed.

- ▶ Accounts payable management

Automate the process of matching documents to invoices through automated data capturing to enable automatic approval routing. As a result, errors are reduced and fraud prevention is enabled by way of background controls.

3.1.2 Optimizing in-transaction decisions at scale with AI on IBM zSystems

The strategy around AI on IBM zSystems is focused on obtaining valuable business insights and operational excellence by using a holistic approach. This strategy includes the following key components:

- ▶ Enables the use of open source data science packages on the platform, such as TensorFlow or PyTorch.
- ▶ Optimizes libraries and compilers for the IBM zSystems architecture and new AI hardware developments, such as the IBM Integrated Accelerator for AI.
- ▶ Uses open source directly or with IBM AI offerings, such as IBM WMLz or IBM Cloud Pak for Data (CP4D).
- ▶ Offers an extensive portfolio of technologies on the platform to enable embedding and integrating AI into client workloads and IBM delivered offerings.

Challenges, such as the timeliness of data availability, irrelevant insights from degraded or incomplete data, a lack of confidence in the accuracy of the data and resulting predictions, increases in costs, and delayed response times, prevents enterprises from fully using their data and capitalizing on new business opportunities. Deploying AI-infused applications on the IBM zSystems platform, where the transactions occur and transactional data originates, presents unique business benefits that can help enterprises overcome these challenges.

A significant amount of business-critical data originates and is stored on the IBM zSystems platform as a result of processing billions of daily transactions. When response time is critical to SLAs, calling off platform for inferencing can introduce intolerable levels of overhead and latency.

Scoring these transactions on the platform results in reduced latency to more easily meet these stringent SLAs. This faster scoring latency is a result of the data gravity of the platform, where it is more efficient to bring the scoring to where the data is rather than move the data to be scored off-platform. Data gravity also enables real-time insights from the most current data in business-critical workloads. With IBM zSystems, an enterprise can infuse AI into every business transaction at scale with high throughput and low latency with minimal application changes.

DL-based fraud models emerged as a way to efficiently analyze transactional data at scale to detect fraudulent patterns. This approach is more effective than traditional fraud models. However, these models are difficult to use within a transaction due to the complexity of the mathematical operations that must be performed by the model during inferencing and because historically these inferencing requests were sent off platform. Enterprises who want to use DL models in transactions can leverage the IBM z16 Integrated Accelerator for AI, which is optimized for processing a DL model's inferencing request.

Also, because business-critical data is on IBM zSystems, data movement across the organization for inferencing can raise concerns about security, trust, and data quality. With real-time in-transaction scoring on IBM zSystems, the need for data movement is reduced. This reduction in data movement also reduces risks around security and trust, which improves data quality by using metadata that might otherwise be lost. It also reduces decision-making time.

The ability to scale AI on IBM zSystems to score every transaction in real time reduces the opportunity for fraudulent behavior, saves costs, increases customer satisfaction, and achieves lower and more consistent response times. You no longer need to take on unwanted risk by inferencing off platform. Furthermore, you have the time to run critical business processes within the transaction, such as fraud detection, without renegeing on your SLAs.

Using a system that allows you to seamlessly embed AI into z/OS applications helps accelerate your enterprise's AI journey and realize positive outcomes sooner. WMLz is a key enablement technology in this system that readily embeds ML and DL models in transactional applications to deliver real-time insights.

WMLz also enables the retention of models with current data, which improves accuracy and trust, and delivers simplified and streamlined model management. It enables inferencing of every transaction with direct calls from transactional workloads to model scoring services with minimal effect on operational processing SLAs for no-compromise AI.

3.1.3 Solution architecture overview

Recall from 1.3, “Building and training anywhere and deploying and infusing on IBM zSystems” on page 8, that you can develop and train ML models by using the platform of your choice. One of the reasons that this task can be done is because of the growing number of supported model formats on IBM zSystems, in particular the Open Neural Network Exchange (ONNX) format.

When deploying a model on IBM zSystems that was developed off-platform, numerous deployment options are available, and of those options, we explore three of the more popular approaches that are optimized for IBM zSystems scoring:

- ▶ Scoring with TensorFlow Serving
- ▶ Scoring with WMLz Online Scoring Community Edition
- ▶ Scoring with IBM Watson Machine Learning for z/OS

Scoring with TensorFlow Serving

The first deployment option uses TensorFlow Serving on IBM zSystems to serve models that are developed with the TensorFlow framework on any platform.

The process flow of in-transaction scoring that uses TensorFlow on IBM zSystems is shown in Figure 3-1 on page 42 and includes the following steps:

1. Receive inbound transaction from a system of engagement.
2. A z/OS CICS application receives control and processes the inbound transaction.
In addition to standard business logic, a credit scoring check is initiated to validate the transaction.
3. The CICS application interacts with the IBM Db2 database, which stores account history, data, and related information to retrieve account data and update records as needed.
4. The CICS application drives a Representational State Transfer (REST) request to the inference service, which is hosted in an IBM z/OS Container Extensions (zCX) instance on the IBM zSystems platform.

5. An online data preparation service receives control to handle any data preprocessing and aggregation that is not implemented as part of the TensorFlow model. This step is *optional*.
6. The model that is hosted in TensorFlow Serving is started to process the prediction and passes the result back through the calling stack. There, the application logic determines how to proceed based on the prediction that is returned.

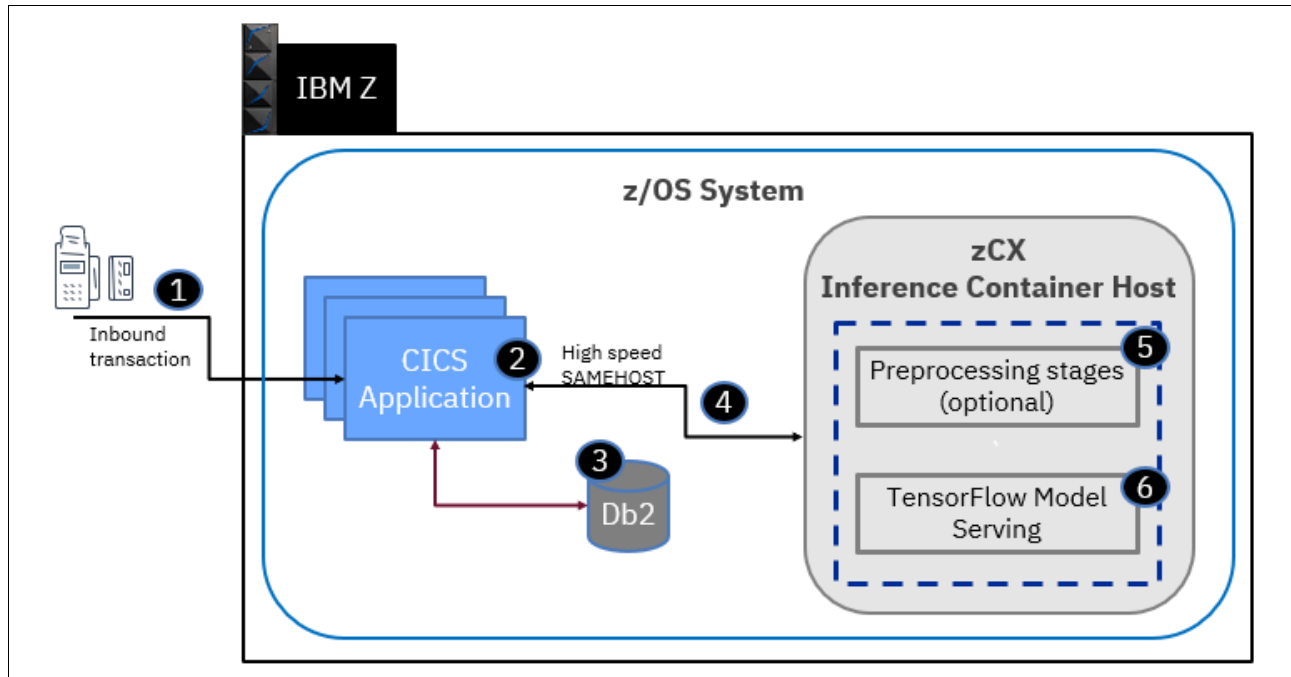


Figure 3-1 Flow of in-transaction scoring with TensorFlow on IBM zSystems

TensorFlow

One major component of this architecture is TensorFlow. TensorFlow is an end-to-end open source platform for ML. It includes a comprehensive, flexible system of tools, libraries, and community resources that allows researchers to push the state-of-the-art in ML and enables developers to easily build and deploy ML infused applications.

It is one of the most widely used open source DL frameworks with AI pipeline management and efficient inferencing capabilities for deep neural networks. With IBM focusing on enabling the use of open source AI frameworks on the IBM zSystems platform, you can use TensorFlow to build, train, and serve ML models, all within the secure high-performance environment of IBM zSystems, where your mission-critical data and applications are stored.

Model building and training are simplified with TensorFlow through its use of intuitive high-level application programming interfaces (APIs), which makes for immediate model iteration and simple debugging. The framework can run on the CPU, GPU, or TPU on servers, desktops, and even mobile devices.

After a trained model is created and ready to be deployed, TensorFlow Serving facilitates deploying models to begin serving prediction requests. This function simplifies and speeds up the process of taking a model into production. You can deploy on different operating systems and platforms, such as cloud, on-premises, in the browser, or on-device, regardless of the programming language that is used.

Opting to use TensorFlow as model serving on IBM zSystems is ideal in the following circumstances:

- ▶ When running multiple models on a large scale.
- ▶ When you require direct support of TensorFlow assets (models, pipelines, and so on).
- ▶ If you want a consistent TensorFlow system experience.
- ▶ When you must configure serving infrastructure to scale.
- ▶ When REST API overhead is acceptable.

Note: TensorFlow Serving can be used for other types of models in addition to TensorFlow models.

Although TensorFlow is not yet ported to z/OS natively, you can still deploy it to your z/OS environment by using zCX. zCX can run Linux on IBM zSystems compatible containers in a z/OS address space, which is a requirement to build TensorFlow by using the TensorFlow Linux on IBM zSystems Docker image. This feature enables you to deploy TensorFlow Docker images directly on zCX in proximity to mission-critical z/OS workloads.

zCX provides the functions to expand and modernize your z/OS software system to include Linux applications, which can leverage your IBM zSystems investment and its renowned qualities of service. For more information about the most up-to-date TensorFlow Linux on IBM zSystems Docker image, see [IBM zSystems Container Image Registry](#).

Scoring with WMLz Online Scoring Community Edition

The second deployment option, and the one we chose to implement for our credit risk model deployment, is using WMLz Online Scoring Community Edition (OSCE) as the scoring engine. The model development and deployment is described in further detail in 3.2, “Credit risk assessment use case scenario” on page 48.

WMLz serves as an end-to-end ML platform for AI on z/OS. It delivers predictive analytics capabilities to the platform and enables the generation of real-time insights at the source. It also provides essential model versions, auditing, and monitoring with high availability (HA), high performance, low latency, and ML model automation or ML as-a-service.

WMLz OSCE offers a no-charge option to test the WMLz feature of real-time inferencing of pre-trained DL ONNX models. Neural network models can be served to z/OS applications through a REST API. For more information about the use of this feature, see 3.2.2, “Credit risk model deployment and evaluation on IBM zSystems” on page 55.

The process flow of in-transaction scoring by using WMLz OSCE is shown in Figure 3-2 and includes the following steps.

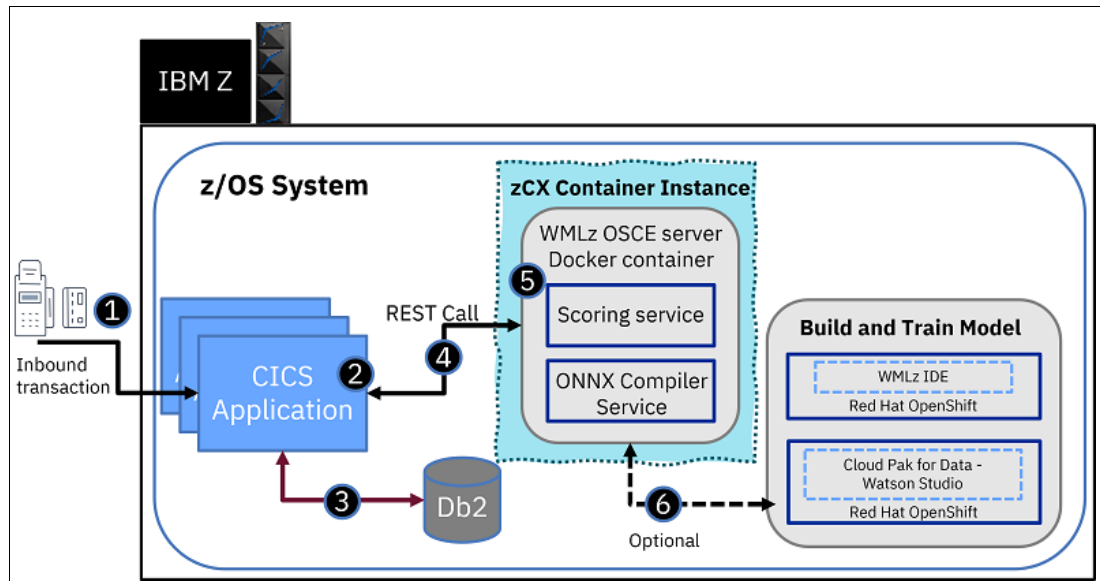


Figure 3-2 Flow of in-transaction scoring with WMLz

1. Receiving an inbound transaction from a system of engagement triggers the scoring process.
2. The z/OS CICS application receives control to process the transaction.
In addition to standard business logic, a credit scoring check starts validation of the transaction.
3. The CICS application communicates with the Db2 database to access and retrieve account history, data, and related information that is necessary to make an accurate prediction and update records as needed.
4. The CICS application triggers a REST API request (or REST call) to the WMLz OSCE server that is inside a zCX instance.
5. The WMLz OSCE scoring service receives the request, starts an instance of the scoring model to process the prediction, and then returns the inference result back through the calling stack to the CICS application, where the application logic can act based on the value that is returned.
6. Optionally, you can use the WMLz integrated development environment (IDE) for model training and development. The WMLz IDE provides various ML modeling tools, run times, and libraries with which data scientists can explore mission-critical data on IBM zSystems and train models to extract actionable insights from that data.

You also can install and use WMLz IDE on Linux on IBM zSystems or x86. Then, you can import those trained models into WMLz for deployment and management.

Alternatively, you can use CP4D with IBM Watson Studio to build ML models for optimized decision-making. You can use CP4D to enable a hybrid cloud approach to AI for accelerated insights across cloud and on-premises sources, improvements in productivity with reduced data integration requests, and cost savings by eliminating manual cataloging. CP4D and Watson Studio run on Red Hat OpenShift on Linux on IBM zSystems, and x86 servers.

For more information about the many fit-for-need CP4D deployment options that are available to expand your AI services, see [IBM Cloud Pak for Data Deployments](#).

Scoring with IBM Watson Machine Learning for z/OS

The third deployment option, WMLz, is used to use our DL credit card fraud model for our second use case. This implementation is described in more detail in 3.3, “Credit card payments fraud detection use case scenario” on page 60.

WMLz delivers predictive analytics capabilities to the platform and enables the generation of real-time insights at the source of a transaction. It offers a flexible model development environment with which you develop on your platform of choice, and deploy models with your transaction services to benefit from low latency, high performance, greater resiliency, and the security rich environment of IBM zSystems. It also provides essential model versions, auditing, and deployment monitoring.

We chose to implement our credit card fraud model in WMLz instead of TensorFlow Serving because of the ease of leveraging the IBM Integrated Accelerator for AI through WMLz. This acceleration is essential for enterprises hoping to leverage AI in high-throughput transactional environments. Another unique feature that WMLz provides is the ability to deploy a model to a z/OS CICS region, which eliminates the RESTful API impact that is present with the TensorFlow Serving implementation.

The process flow for our credit card fraud deployment, which is shown in Figure 3-3, includes the following high-level steps.

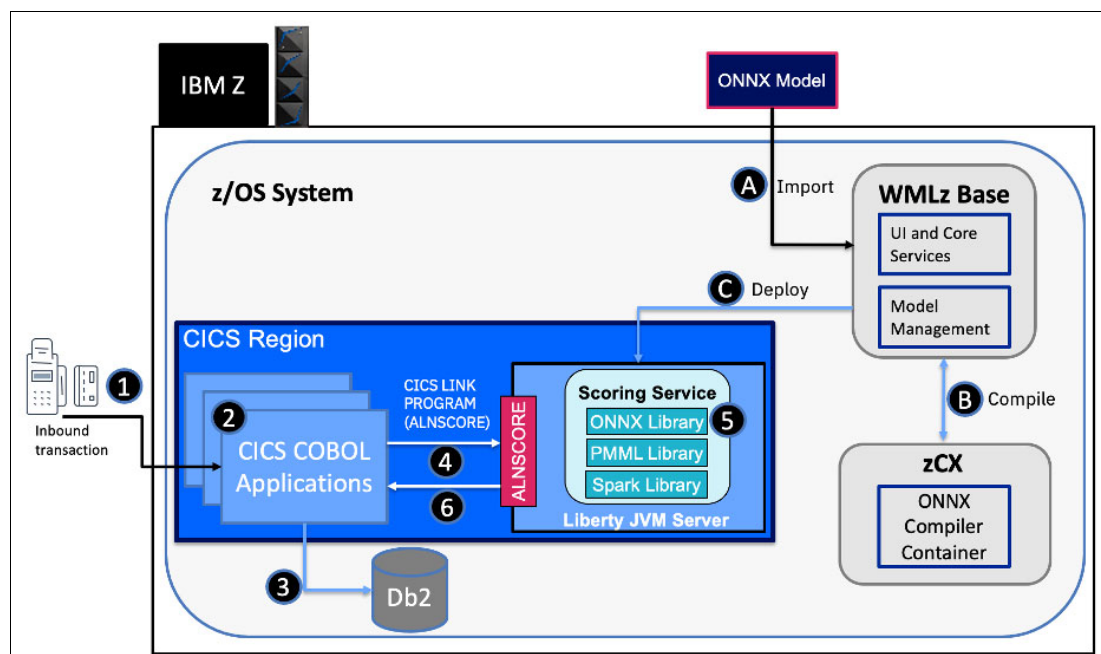


Figure 3-3 WMLz ONNX deployment and score process flow

1. Import: When a TensorFlow Deep Learning credit card fraud model is trained by a data scientist, it is converted to the ONNX model format and imported into WMLz.
2. Compile: When the model is imported into WMLz, it is compiled by the IBM zSystems Deep Learning Compiler (zDLC), which is run within a container in a zCX instance. The ONNX model is now optimized for the IBM Integrated Accelerator for AI and transferred to WMLz for storage in its metadata repository.
3. Deploy: The compiled model is deployed to a scoring server that is running within a Liberty Java virtual machine (JVM) in a CICS region. The model is now ready to complete inferencing requests that are sent to it by CICS applications.

These steps are described in more detail in 3.3.2, “Fraud detection model deployment and evaluation on IBM zSystems” on page 66.

Also shown in Figure 3-3 on page 45 is the process flow of inferencing a credit card transaction for potential fraud within the transaction. The steps are as follows:

1. An inbound transaction is received by z/OS from the point-of-sale system.
2. The z/OS CICS COBOL application that is responsible for processing the credit card transaction takes control.
3. The CICS application communicates with the Db2 database to access and retrieve account transaction history, data, and related information that is necessary to make a fraud prediction and update records as needed.
4. The CICS application triggers the fraud scoring call by using a WMLz provided program that is called ALNSCORE. A **CICS LINK** command is used, and a **CICS PUT** command is used to transfer the model inputs from the COBOL program to the ALNSCORE program.
The ALNSCORE program passes along the input data to the ONNX scoring service.
5. The ONNX scoring service within the Liberty JVM performs the optimized inferencing request leveraging the IBM Integrated Accelerator for AI.
The fraud prediction result is returned to ALNSCORE.
6. A **CICS GET** command is used to retrieve the outputs from the inferencing request for this transaction. The CICS program has an indication whether the current customer transaction is fraudulent or not and can either deny or complete the transaction.

In addition to the components that previously were described (TensorFlow Serving, zCX, CP4D, WMLz OSCE, and WMLz), the high-level solution architecture for in-transaction scoring on IBM zSystems might include the following components and integrations (described in the corresponding sections):

- ▶ IBM CICS Transactions Server for z/OS
- ▶ IBM Db2 database for z/OS
- ▶ Docker containers

IBM CICS Transactions Server for z/OS

Starting with our sample application, IBM CICS Transactions Server for z/OS (CICS) is a mixed-language application server that runs on the IBM zSystems platform. An *application server* provides the environment to host applications.

CICS is used as an application server by many enterprises in various industries, especially by enterprises in the financial services sector that require high levels of security, scalability, and cost efficiency for processing billions of high throughput workloads, such as credit card transactions, across the world daily.

With its mixed-language capability, CICS can host applications that are written in many different languages (from Java EE to COBOL), and these mixed-language applications can share core programming concepts. This feature enables developers to write applications in the best language for the requirements and use modern languages without the rework of existing applications. The IBM CICS program logic manages the following tasks:

- ▶ Creates the scoring input record.
- ▶ Starts the REST call to the credit risk model for scoring.
- ▶ Prints the scoring result to the CICS region.
- ▶ Reads preprocessed credit application data from the input.
- ▶ Prints the score for each application to an output file.

For more information about the application logic for our sample CICS application, see 3.2.3, “CICS application integration for real-time credit risk scoring” on page 58.

IBM Db2 database for z/OS

Because your model might use data that is not normally part of the transaction, this data must be collected. An example of this type of information might be more information about your customer that is based on the customer’s ID. This customer data might be personal or historical data that was saved in a database.

The database that is used in the architecture that is shown in Figure 3-2 on page 44 is the IBM Db2 database for z/OS. This database is a highly scalable, reliable, and cost-effective relational database management system that runs on the platform.

A relational database is a database in which all of the data is logically contained in tables and organized according to the relational model. By using Db2 for z/OS, data can be defined and manipulated by using Structured Query Language (SQL), which is the standard language for accessing data in relational databases.

Db2 for z/OS provides the functions that are needed to handle rapid changes in diverse and unpredictable workloads while providing optimal resource usage and investment. The database stores the historical and real-time data that is required to make accurate predictions. The tight integration that Db2 for z/OS has with the IBM zSystems architecture and the z/OS environment creates a synergy that allows Db2 to use advanced z/OS functions.

Docker containers

To enable WMLz OSCE in a z/OS environment, you must enable and configure z/OS Container Extensions. Also, you must provision a zCX instance to host the WMLz OSCE server that is running as a Docker container. The WMLz OSCE server Docker container is composed of the web user interface, the ONNX scoring service, and the ONNX compiler service. Additionally, WMLz requires a zCX instance only for the ONNX compiler service, which runs within a container.

Deploying the container in a zCX instance makes it possible to modify the deployment scenario to deploy to a Linux on IBM zSystems system, if needed.

Many AI libraries and frameworks support the functions of exporting or converting a trained model to ONNX format. After a model is in an ONNX format, it is portable to any platform without any runtime dependencies that are based on the libraries or framework on which it was trained.

As part of the import process (see “Deploying the model with WMLz OSCE” on page 55), the ONNX compiler service compiles our model into executable code that is optimized for the IBM zSystems platform for scoring with the help of the zDLC. The ONNX scoring service handles the REST request from the application, starts the inference process, and returns the scoring result to the application.

A REST API, also known as *RESTful API*, is an API that conforms to the constraints of the REST architectural style and allows for interaction with RESTful web services. An *API* is a set of definitions and protocols for building and integrating application software. You can think of an API as a mediator between the users or customers, and the resources or web services that they want to get. It is also a way for an organization to share resources and information while maintaining security, control, and authentication, which determines who can access specific information.

3.2 Credit risk assessment use case scenario

Current credit risk assessment processes tend to be largely manual, resource-intensive, and untrustworthy due to a lack of transparency around decision making, and they cannot use the high volumes of complex data from various sources at rest or in motion.

Enterprises need an automated solution that provides explainable AI to increase customer trust and protect themselves from potential liability. They must do so while maintaining increasingly strict regulatory compliance requirements and meet stringent SLAs with agility and flexibility. Through infusing AI into credit risk assessment applications with ML models, financial institutions can do just that.

Our use case demonstrates how we easily infuse AI into an IBM CICS application that is on IBM zSystems by using a TensorFlow ML model that is started by a REST API call. Additionally, that model is deployed with WMLz OSCE and used by providing ML-based credit risk scoring within a loan application.

We also briefly describe the same scenario by using TensorFlow as the scoring engine instead of WMLz. The purpose of our example application is to start the deployed model, use financial data to make trusted predictions about an applicant's default risk, and present that risk assessment to the user. This assessment result serves as the foundation for credit or loan approval or denial and the reasoning behind the decision.

When a customer applies for a loan, the lending institution must evaluate whether the applicant can reliably repay the principal loan amount and any accrued interest. Lenders or banks commonly use measures of profitability and leverage to assess credit risk.

The concept of leverage helps determine whether the applicant is a risk or an investment that is worth making. When referring to a high leverage applicant, it is understood that the applicant has more debt than equity, which increases the risk level of the applicant.

Given two loan applicants (one with high profitability and high leverage, and the other with low profitability and low leverage), which applicant has lower credit risk?

The complexity of answering this question multiplies when banks incorporate the many other dimensions that they examine during the credit risk assessment. These other dimensions typically include other financial information, such as liquidity ratio, or behavioral information, such as loan or trade credit payment behavior to gain a more holistic view of the specific applicant.

Summarizing all these various dimensions into one predictive score is challenging, but ML techniques help achieve this goal with speed and a customer focus.

3.2.1 Data science and credit risk model development

Now that we have a better understanding of the architecture that supports and enables our use case implementation, we provide an overview of the data science process of our neural network credit risk model development. The goal of the process is to create a simple neural network that predicts whether approving a loan for a customer poses a risk.

Understanding and preparing the data

When building a successful AI solution, your insights and analytics are only as good as the data. Therefore, it is critical that understanding the data becomes a team effort. That team includes (but it not limited to) members with the following expertise:

- ▶ Data scientists
- ▶ Business analysts
- ▶ Subject matter experts
- ▶ Data stewards

In our implementation, we use a prepared synthetic credit risk data set of 5000 customers that was obtained from the UCI Machine Learning Repository¹. For more information, see [UCI Machine Learning Repository: Statlog](#).

To help better understand this data, we document some data descriptors that are included in the data set. The following data aspects are included in our data set:

- ▶ Customer ID
- ▶ Seven-integer valued variables
- ▶ A total of 13 categorical features that are represented by strings
- ▶ Risk label that is represented by strings

By using a Google Colaboratory Notebook, complete the following steps:

1. Eliminate the CustomerID from the data set because it features missing values.
2. Encode all categorical values and our Risk label where No Risk = 0 and Risk = 1.
3. Save the encoding in our data dictionary, as shown in Example 3-1.

Google Colaboratory is a hosted Jupyter Notebook service (called Colab Notebook), which was developed by Google Research to write and run Python code in a web browser.

Example 3-1 Encoding the variables with Google Colab

```
#remove CustomerID as it holds no value for our model
df = df.drop(columns=['CustomerID'], axis=1)

#encode the label
df = df.replace({'Risk': {'No Risk': 0, 'Risk': 1}})

#encode categorical values
dictionaries = dict()
for col in df:
    if df[col].dtype != 'int64':
        unique = df[col].unique()
        unique_size = unique.size
        dictionary = dict(zip(unique, range(unique_size)))
        df = df.replace({col:dictionary})
        dictionaries[col] = dictionary

#save the encoding
np.save('/content/dictionaries.npy', dictionaries)
```

¹ Hofmann, Hans. (1994). Statlog (German Credit Data). UCI Machine Learning Repository.

The first thing that must be reviewed in the data set (before encoding is done) is the distribution of the variables, starting with the Risk label that is shown in Example 3-2. This review is done to familiarize ourselves with the data set.

Example 3-2 Distribution of the target variable

| | |
|---------|------|
| Risk | |
| No Risk | 3330 |
| Risk | 1670 |

When reviewing the explanatory variables in your data set, pay attention to the ranges for which you have data. Histograms provide a valuable method to visualize your data and observe the distribution of variables within the data set.

Data is considered useful when it is evenly distributed among all potential values for each feature. In our example, we can see from the histograms that are formulated from our data set in Figure 3-4 that a Loan Duration for up to 36 months and a Loan Amount of less than \$7,500 are represented in our data set.

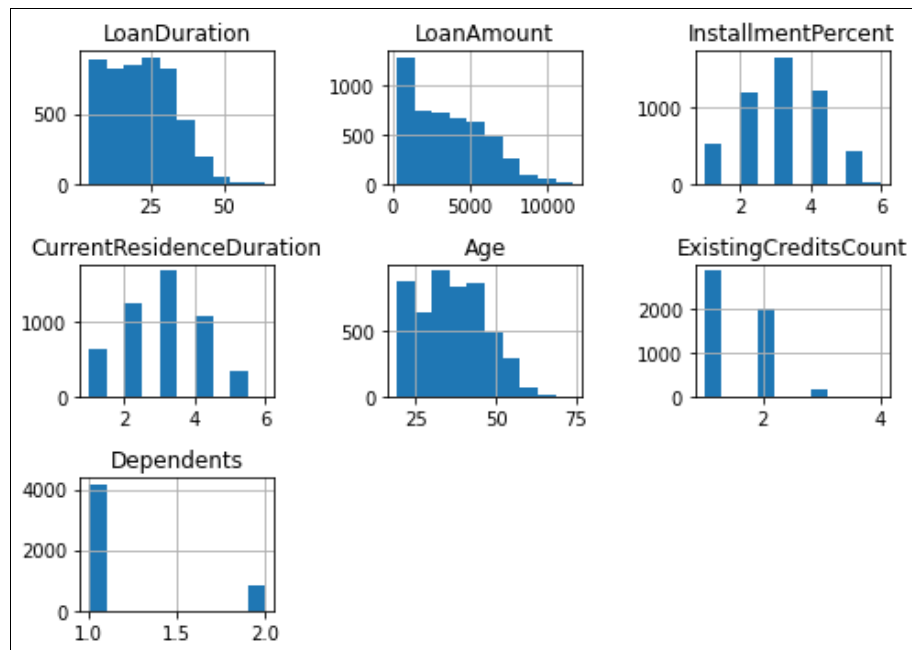


Figure 3-4 Distribution of the numerical values

However, longer durations are less represented. By using this information, we conclude that the resulting model from this data provides accurate scores for loans that amount to less than US \$7,500 and have a duration of less than 36 months. That is, the model is best suited for shorter term, smaller value loans. This information also indicates that the same model provides less accurate scores for longer term, larger value loans, such as a 10-year term loan for US \$50,000.

Similarly, we can see that little representative data exists for people over the age of 60; therefore, the usage of the model to make a prediction about someone of that age or older might result in poor performance and inaccurate results. For transparency, this type of analysis must be documented with the model so that the users are informed about its limitations.

Finally, we review the categorical data variables, which are shown in Example 3-3. We can easily obtain the number of occurrences of each category in our data set with the pandas function, `value_counts`, which is applied to each column separately. In doing so, we can see that almost all our data points are from foreign workers. This information tells us that the model might not perform well for non-foreign workers.

Example 3-3 Distribution of the categorical variables

| | |
|------------------------|------------------------------|
| CheckingStatus | CreditHistory |
| 0_to_200 1304 | all_credits_paid_back 769 |
| greater_200 305 | credits_paid_to_date 1490 |
| less_0 1398 | no_credits 117 |
| no_checking 1993 | outstanding_credit 938 |
| | prior_payments_delayed 1686 |
| LoanPurpose | ExistingSavings |
| appliances 561 | 100_to_500 1133 |
| business 146 | 500_to_1000 1078 |
| car_new 945 | greater_1000 558 |
| car_used 808 | less_100 1856 |
| education 167 | unknown 375 |
| furniture 853 | EmploymentDuration |
| other 113 | 1_to_4 1470 |
| radio_tv 755 | 4_to_7 1400 |
| repairs 283 | greater_7 930 |
| retraining 164 | less_1 904 |
| vacation 205 | unemployed 296 |
| Sex | OthersOnLoan |
| female 1896 | co-applicant 717 |
| male 3104 | guarantor 110 |
| | none 4173 |
| OwnsProperty | InstallmentPlans |
| car_other 1540 | bank 466 |
| real_estate 1087 | none 3517 |
| savings_insurance 1660 | stores 1017 |
| unknown 713 | |
| Housing | Job |
| free 739 | management_self-employed 641 |
| own 3195 | skilled 3400 |
| rent 1066 | unemployed 286 |
| | unskilled 673 |
| Telephone | ForeignWorker |
| none 2941 | no 123 |
| yes 2059 | yes 4877 |

After becoming familiar with our data, you can think about the business problem and credit risk, and consider how you can use the data to answer the business problem. You must predict how much of a credit risk the customer poses. This data understanding and preparation work helps prevent rework in the next stage: model creation.

Selecting the credit risk model

For our real-time credit scoring implementation, we use a neural network DL model for the following reasons:

- ▶ Neural networks are capable models that can be used in various circumstances.
- ▶ We can easily transport this model type by using the ONNX format.

DL neural networks are mathematical models for representing data. They allow computer programs to recognize patterns and solve common problems in the field of AI, ML, and DL.

The name and structure of neural networks are inspired by the human brain and mimics neurons. Neural networks are composed of node layers, input layers, and one or more hidden layers. These node layers consists of multiple layers of neurons, which hold specific weights that are trained or updated when data is provided at the input.

Each node connects to another and has an associated weight and threshold. Nodes are activated by producing an output that is greater than the specified threshold, which then sends data to the next layer of nodes. Neural networks rely on training data to learn and improve their accuracy over time and represent the data more accurately when a large amount of data is used for training.

Neural networks use an activation function to get the output of a node. Two types of activation functions are available:

- ▶ Linear: When an activation function is linear, no output range exists because the functions range is unlimited.
- ▶ Nonlinear: These functions are the most commonly used activation functions. They make it easy for a model to adapt to various data and differentiate between the output.

For our model, we use two activation functions: a sigmoid activation function, and a Rectified Linear Unit (ReLU) activation function. The sigmoid activation function resembles an 'S' shape and is a nonlinear activation function, which exists between (0,1). The ReLU activation function is another nonlinear activation function and is mostly used on convolutional neural networks with a range of (0, infinity).

Consider each individual node as its own linear regression model. The node is composed of input data, weights, a bias (threshold), and an output. The way that the nodes are composed is with an input layer for the training data to enter; then, the deep neural network that is made up of multiple hidden layers has the nodes communicating with each other and passing data to the next layer until the data reaches the dropout layer to produce an output.

Training the credit risk model

Many frameworks are available to develop and train neural networks, such as PyTorch or TensorFlow. In our implementation, we use TensorFlow because of the familiarity of the framework, and its use of Keras libraries to build neural networks.

One of the advantages of the "Train anywhere and deploy on IBM zSystems" approach is that you can choose the development environment that best suits your needs. In our example, Google Colab is the training environment of choice. It is a web-based development environment that runs on Python, uses Jupyter Notebooks, and does not require any installation.

Because Google Colab is web-based, the code and run time can be easily shared with all the members of our team regardless of their geographic location, which enables better global collaboration.

The first step in model development is to load the data into the Google Colab Notebook by using the following command and replacing **URL to CSV file** with our specific URL to the following data file:

```
df = pandas.read_csv("URL to CSV file")
```

Eliminate the customer ID and encode the categorical data, as shown in Example 3-1 on page 49.

In our example, we chose to use a simple encoding and assign numbers to each of the categories. If we had more time and resources, other encoding options are available: one hot encoding, or a more thoughtful encoding into numbers.

We must get the data prepared for use to build and train the model, as shown in Example 3-4.

Example 3-4 Shuffling, splitting, and saving the data

```
#shuffle the data
df = df.sample(frac=1, random_state=123)

#split into train and test data
train, validate, test = \
    np.split(df.sample(frac=1, random_state=123),
            [int(.8*len(df)), int(.9*len(df))])

#save the data splits
train.to_csv('/content/german_credit_data_num_train.csv', index=False)
validate.to_csv('/content/german_credit_data_num_validate.csv', index=False)
test.to_csv('/content/german_credit_data_num_test.csv', index=False)
```

Shuffle the data, split it into training, validate it, and test the data by using the 80/10/10 model in which 80% of the data is dedicated to training, 10% is for validation, and 10% is for testing. Then, the data splits are saved. For the training data, split the target variable into separate data frames.

We are ready to start building the model. Start with creating a normalization layer, `normalizer`, in the Notebook:

```
normalizer = tf.keras.layers.Normalization(axis=-1)
```

This normalization technique helps ensure that the scale of the values is balanced and the range of the values are maintained and proportional because our numerical values were on different scales. After creating the normalization layer, adapt the normalization data to our data by using the following command:

```
normalizer.adapt(x_train)
```

After we adapt the normalization layer, we can create a simple neural network in the Notebook with `normalizer` as our input. After we create the model with the input, we can create our first dense, or hidden, layer with 64 neurons, and the activation function ReLU. The commands that support the creation of our neural network with `normalizer` as the input, the first dense layer, and the activation function are shown in Example 3-5. ReLU is the most commonly used activation function, and when we train a production-ready model, we must select the optimum activation function based on tests.

Example 3-5 Commands to create a simple neural network with normalizer as input

```
model = Sequential()  
model.add(normalizer)  
model.add(Dense(64, activation='relu'))
```

A dense layer includes every neuron that is connected to every input from the previous layer. In our case, all 64 neurons take all 20 features as input. To improve the performance of our model, we apply a dropout layer of 50% by using the following command:

```
model.add(Dropout(0.5))
```

During training, this dropout layer randomly sets inputs from the previous dense layer to zero. The remaining inputs are scaled so that the sum of the inputs remains the same. The idea behind dropout is to prevent over-fitting. After we apply the first dropout layer, we add another dense layer with 64 neurons and another dropout layer with 50% dropout, and another dense layer with 64 neurons by using the following commands:

```
model.add(Dense(64, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(64, activation='relu'))
```

Finally, a dense layer exists with a single neuron and the sigmoid activation function is added by using the following command:

```
model.add(Dense(1, activation='sigmoid'))
```

We use the sigmoid activation function because the function exists 0 - 1. Because we are predicting probability, we are looking for output of zero or one. This neuron takes all neurons from our previous layer as inputs and outputs a value between zero and one because of the sigmoid function.

In our model, zero means no risk and one means risk. Therefore, a high value, such as 0.91, indicates that the model suspects risk; a low value, such as 0.07, indicates that the model does *not* suspect the credit applicant to be a risk.

After creating the model, we must configure it for training. To do so, select an optimizer (`adam`), loss function (`tf.keras.losses.BinaryCrossentropy()`), and our metric for success (`accuracy`) by using the following commands:

```
model.compile(optimizer='adam', loss=tf.keras.losses.BinaryCrossentropy(),  
metrics=['accuracy'])
```

Now, we can train our model by using our training data while also providing data for validation by using the following command:

```
model.fit(x_train,y_train, validation_data=(x_val, y_val), epochs=40)
```

After we train our model, convert it to an ONNX model format and save it to a file with the help of the `tf2onnx` library by using the following command:

```
tf2onnx.convert.from_keras(model, output_path='/content/credit-risk-num.onnx')
```

Finally, we are ready to deploy our model on IBM zSystems.

For more information about the full Notebook, see Appendix B, “Additional material” on page 105.

3.2.2 Credit risk model deployment and evaluation on IBM zSystems

Deploying a model means to take the built and tested model and integrate it with other business applications. By deploying the model, it is made available for consumption across the enterprise. After we deploy a model, we can score the model to generate predictions, evaluate it, and retrain it as needed. To deploy our neural network credit risk model, we use WMLz OSCE and the APIs that it provides to store, deploy, update, and delete models.

Deploying the model with WMLz OSCE

To enable a simple and smooth deployment of our model, we use the WMLz OSCE User Interface (UI). Before the model can be deployed, we first install and configure the OSCE service and obtain access to the OSCE UI. Because we must allow multi-user access to the OSCE UI, we note the full URL of the OSCE UI and the access token to share with the team after the OSCE server successfully starts:

OSCE UI URL: `http://129.40.23.87:8080/#/models/2`

Access token: `5301acb9-af3e-4676-a601-a0020f2b5c2a`

After the OSCE is running, complete the following steps to import the ONNX model by using the OSCE UI:

1. Log in to the OSCE UI, shown in Figure 3-5, by using the access token that was generated during the OSCE installation and configuration, `5301acb9-af3e-4676-a601-a0020f2b5c2a`.

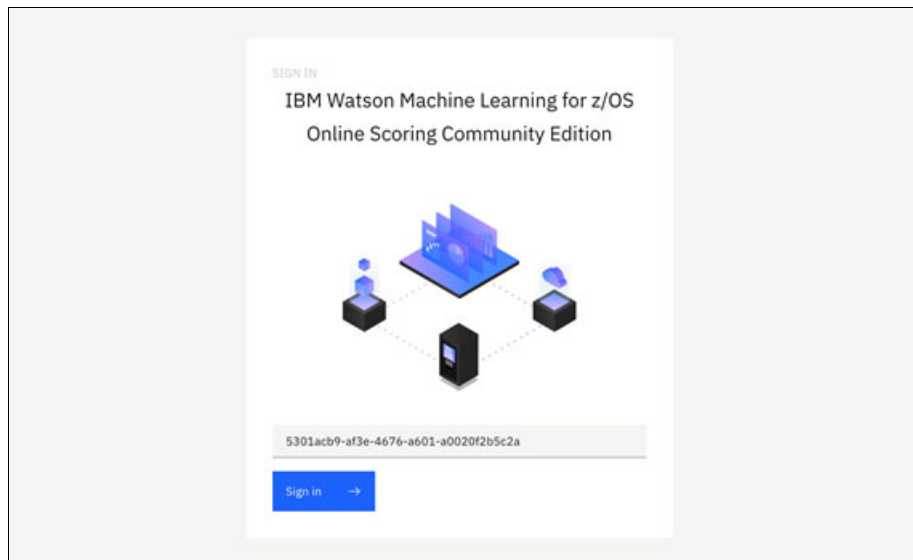


Figure 3-5 WMLz OSCE UI login

2. Select the **Models** tab and click **Import Model**, as shown in Figure 3-6.

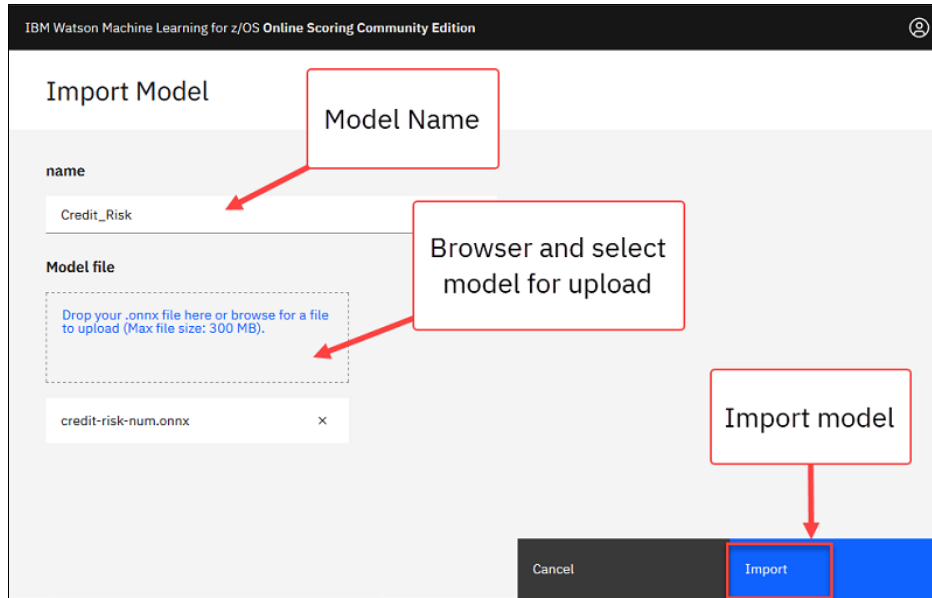


Figure 3-6 Importing a model into WMLz OSCE UI

3. The Import Model page opens, in which the model name can be specified:

Credit_Risk

4. Upload the model source file (credit-risk-num.onnx) which we can see is in the required ONNX format by its .onnx file name extension.

The model source file can be uploaded by using the file browser method, or by dropping the file into the Upload field.

5. After the model source file is selected, click **Import**, and wait for a successful compile to complete. A valid status is shown on the Models page. This process can take some time, depending on the size of the model.

After the model is imported, the ONNX model compiler and the zDLC compile it. This process can be a time-consuming task depending on the size and complexity of the model that is uploaded. For our example, the model upload and zDLC compilation process completed quickly because of the simplicity of our model.

After the import and compile process is complete, the status (Deployed) appears, and includes a green checkmark next to it (see Figure 3-7).

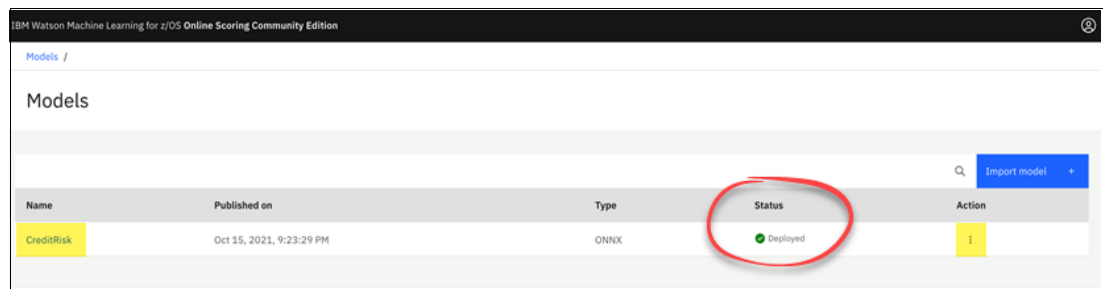


Figure 3-7 Model shows that it successfully deployed

Some useful information now can be gathered about the model and tested by clicking **CreditRisk** (see the Name column in Figure 3-7 on page 56), or by clicking the three stacked dots menu (see the Action column in Figure 3-7 on page 56).

When you select the **General** tab, you can see the URL for the scoring endpoint (we use REST calls to this URL to score our model). Select the **Schema** tab to show the format of the input and output that the model expects. Finally, select the **Test** tab to run test cases against the model.

Testing the model with WMLz OSCE

When testing a model, you can score one or more inputs simultaneously by enclosing all the inputs in square brackets and separating them by using commas. For our example, we test score on a single record from our test data, which the model has not yet seen. After the test data is entered for the model, click **Submit** to score it, as shown in Figure 3-8.

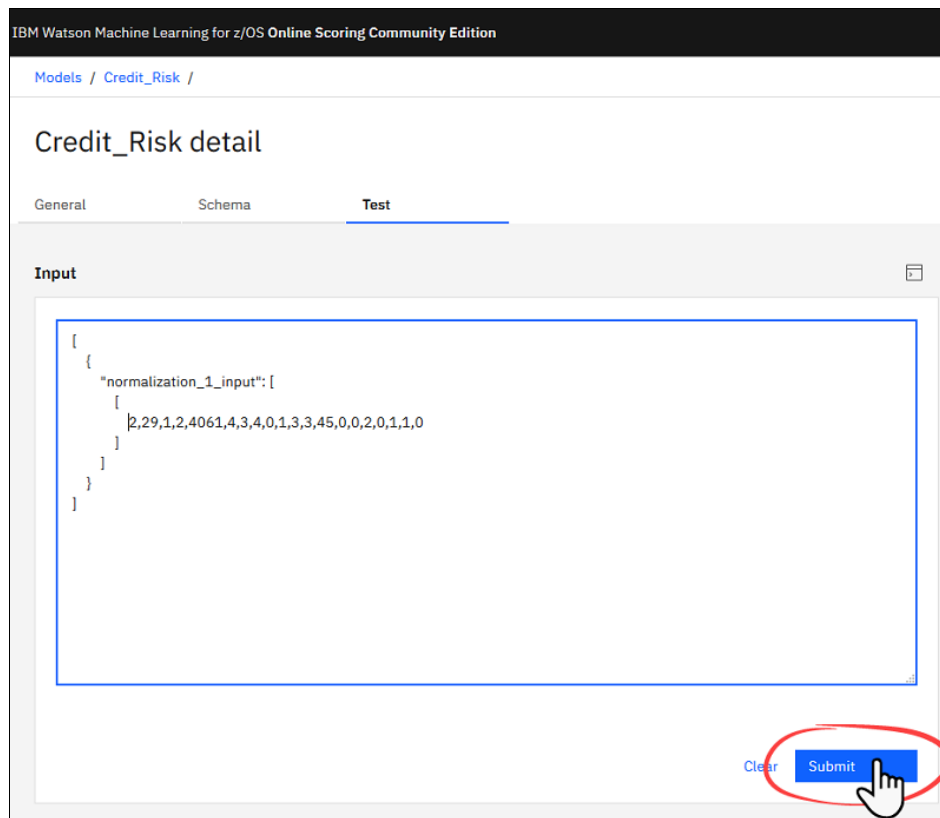


Figure 3-8 Test model input in WMLz OSCE

After submitting the test data to the model for scoring, you can view the returned results, including a description of the returned data and the data that is available (see Figure 3-9). In general, when interpreting the resulting probability score that predicts whether an applicant is risky, the closer a risk probability score is to 1 than 0, the greater the likelihood that the applicant presents a high credit risk. Conversely, the closer a risk probability score is to 0 than 1, the greater the likelihood that the applicant presents a low credit risk. Scoring results are likely to be broken down into multiple risk levels to more accurately represent the business case. In our example, we received a score of 0.9678754210472107, which means that our model predicts that providing credit to this applicant presents risk.

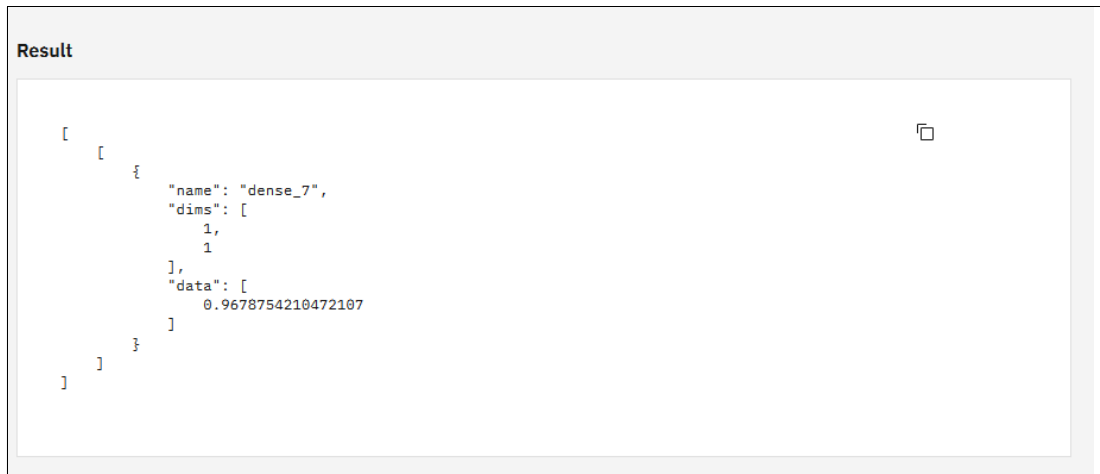


Figure 3-9 Viewing the test model result in WMLz OSCE

Next, call this model by using the REST API. For the sake of simplicity, use a `curl` command:

```
curl \
-X POST http://129.40.23.87:8080/v1/api/models/3/prediction \
-H 'Authorization: 5301acb9-af3e-4676-a601-a0020f2b5c2a' \
-H 'Content-Type: application/json' \
-d '[{"normalization_1_input": [[2,29,1,2,4061,4,3,4,0,1,3,3,45,0,0,2,0,1,1,0]]}]'
```

The result is the same return as the result from testing in WMLz OSCE, shown in Figure 3-9:

```
[{"name": "dense_7", "dims": [1, 1], "data": [0.9678754210472107]}]
```

Similarly, the model can be accessed from any program that supports HTTP calls.

3.2.3 CICS application integration for real-time credit risk scoring

In-transaction scoring that is integrated into a CICS application is about the usage of the real-time scoring model to calculate a risk score for the available data in real time inside the transaction. As shown in Figure 3-10 on page 59, we use CICS Region PLX7CIC1 and create and install all the resources and assets under CICS GROUP ALNGROUP.


```

ENTER COMMANDS
NAME          TYPE          GROUP
CHURNMLP     PROGRAM      ALNGROUP
CHURNSCR     PROGRAM      ALNGROUP
CREDRSCR     PROGRAM      ALNGROUP
DFHOWBC1    PROGRAM      ALNGROUP
DFHOWBC2    PROGRAM      ALNGROUP
SCORECAL     PROGRAM      ALNGROUP
TESTCOBL    PROGRAM      ALNGROUP
CRED         TRANSACTION  ALNGROUP
MLPC        TRANSACTION  ALNGROUP
SCCA        TRANSACTION  ALNGROUP
SCRC        TRANSACTION  ALNGROUP
TEST        TRANSACTION  ALNGROUP
CREDRURI    URIMAP       ALNGROUP
SCOREURI    URIMAP       ALNGROUP

```

Figure 3-10 Installed resources and assets in ALNGROUP

The sample application uses the following CICS resources and assets:

- ▶ PROGRAM(CREDRSCR): The credit risk score prediction program. This program reads preprocessed customer credit application data from an input file, calls PROGRAM(DFH0WBC2) for scoring service, and prints the scoring result for each application to the CICS region on an output file.
- ▶ PROGRAM(DFH0WBC2): This program makes the REST API call to the credit risk scoring model that is deployed on WMLz OSCE.
- ▶ TRANSACTION(CRED): This transaction calls the COBOL program CREDRSCR to demonstrate the capability of calling our Credit_Risk model scoring in real time within a CICS transaction.
- ▶ CREDRURI - URIMAP(CREDRURI): This item is defined with the use type CLIENT, and it points to the Credit_Risk model that is deployed to the WMLz OSCE server. This URIMAP definition specifies the URL `http://129.40.23.87:8080/#/models/2`, which is used when the CICS application makes a request to the HTTP OSCE server, which points to our deployed model.

To test the model for real-time in-transaction scoring by using our CICS application, run the transaction that calls the credit risk score prediction program to start the inferencing process. To start a CICS transaction, press the CLEAR or DELETE key to clear the window. Then, enter the transaction identifier (by itself or followed by data) on the CLI of the window. For our example, run the CRED transaction to call the CREDRSCR program, as shown in Figure 3-11. At the completion of the transaction, the STATUS message of SESSION ENDED appears.

```

CRED
STATUS:  SESSION ENDED

```

Figure 3-11 Running TRANSACTION(CRED)

Now, you can check the scoring input and output in the CICS region address space log (see Figure 3-12) and interpret the results.

```
SDFS OUTPUT DISPLAY PLX7CIC1 STC00703 DSID 108 LINE 1,487 COLUMNS 02- 161
COMMAND INPUT ==> SCROLL ==> CSR
TC11CRED 20211018015705 PERFORM SCORING...
TC11CRED 20211018015705 SCORING INPUT...
TC11CRED 20211018015705 {"normalization_input": 2,61,2,4,7646,3,2,6,1,1,3,2,49,0,1,2,0,2,1,0 }
TC11CRED 20211018015705 SCORING OUTPUT...
TC11CRED 20211018015705 {"name":"dense_3","dims": 1,1 ,"data": 0.9606726765632629 }
***** BOTTOM OF DATA *****
```

Figure 3-12 Scoring input and output in CICS region address space log

The result from our real-time in-transaction scoring is a risk score of 0.9606726765632629, which indicates that this applicant shows risk and might not be an ideal borrower for the lending institution. This result helps guide credit approval decisions that are based on business logic.

For more information about the source code for our sample CICS application and the resources that are used, see Appendix B, “Additional material” on page 105.

3.3 Credit card payments fraud detection use case scenario

Detecting credit card fraud is an area of great importance for credit card companies, payment processing companies, and banks. According to a Nilson Report, in 2020 alone, companies worldwide collectively lost \$28.58 billion US Dollars to card fraud. With the explosion of e-commerce in recent years, in part due to the COVID-19 pandemic, “card-not-present” transactions have increased, making it the majority of credit card fraud and contributing to the ongoing upwards trend of high industry losses due to fraud.²

Preventing fraudulent transactions is paramount to maintaining trust and improving customer experience. Therefore, payment processing companies and banks have fraud prevention systems in place to help detect and prevent fraud. These systems contain a myriad of AI models and rule-based methods to detect fraud. While building and implementing fraud detection systems, keep the false positive rate, for example, flagging non-fraudulent transactions as fraud to a minimum because it creates a negative user experience for loyal customers. By leveraging AI methods in fraud prevention systems, you can enrich the existing rule-based engine approach to fraud detection by increasing the accuracy of models through reduced false positive and false negatives.

As fraud prevention systems are evolving and deploying advanced AI methods, fraudsters are getting smarter. Therefore, IT companies must stay on top of the latest fraud methods and use that information to build better fraud prevention systems in real time.

Transaction duration is a key metric to keep in mind when implementing a fraud prevention system because it can negatively impact the customer experience if it takes too long to process transactions. For example, if payment processing is taking place on IBM zSystems, it makes sense to deploy the fraud prevention model on IBM zSystems too. This action saves time and reduces the risk that is involved in moving data from IBM zSystems to an external system for inference. Such a scenario was not feasible a few years ago, but thanks to the functions that are offered by WMLz combined with the latest hardware upgrades of IBM z16, such as the IBM Integrated Accelerator for AI, it is now feasible to facilitate in-transaction AI inferencing in real time with optimal inferencing performance.

² https://nilsonreport.com/upload/content_promo/NilsonReport_Issue1209.pdf

In this section, we demonstrate how an in-transaction fraud detection and prevention scenario can be implemented by using a DL model that is served by WMLz running on IBM z16. Using a similar approach with our credit risk assessment use case, we demonstrate how we can easily implement a CICS application that infuses an AI-based fraud detection algorithm into an online transaction. Unlike in the previous use case, the AI model is deployed to WMLz, instead of the more limited WMLz OSCE version. As before, a REST API endpoint is created for serving the model, but additional WMLz deployment features are leveraged.

The scenario is based on a large, labeled transactional data set that features samples of fraudulent transactions and legitimate transactions. We attempt to demonstrate a relationship between the legitimacy of a customer's current transaction based on their historical transactions. This relationship is modeled by using a DL algorithm. We provide details about the training data and how the model is trained, which are followed by more details about the deployment of the model to WMLz and the usage of RESTful APIs to serve the model.

For more information and a deeper description about the business case for fraud detection solutions, see 4.1.1, "Business-related use case scenarios" on page 82.

3.3.1 Data science and fraud detection model development

The first step toward building a DL model for detecting fraudulent credit card transaction is to obtain the data that has sufficient amount of fraudulent and non-fraudulent transactions to train the model. After we explain the data, we go through the modeling approach.

Understanding and preparing the data

The data that we used for demonstrating the use case includes credit card transactions. There are 29,757 fraudulent transactions out of 24,386,901 total transactions from 20,000 users. To access and download this sample data set, go to the following [IBM GitHub repository](#).

The fields of the data table (see Figure 3-13) include the following ones:

- ▶ User
- ▶ Card
- ▶ Year
- ▶ Month
- ▶ Day
- ▶ Time
- ▶ Amount
- ▶ Use Chip
- ▶ Merchant Name
- ▶ Merchant City
- ▶ Merchant State
- ▶ Zip
- ▶ MCC
- ▶ Errors?
- ▶ Is Fraud?

| User | Card | Year | Month | Day | Time | Amount | Use Chip | Merchant Name | Merchant City | Merchant State | Zip | MCC | Errors? | Is Fraud? |
|------|------|------|-------|-----|-------|----------|-------------------|---------------|---------------|----------------|-------|------|---------|-----------|
| 0 | 0 | 2002 | 9 | 1 | 6:21 | \$134.09 | Swipe Transaction | 3.52721E+18 | La Verne | CA | 91750 | 5300 | | No |
| 0 | 0 | 2002 | 9 | 1 | 6:42 | \$38.48 | Swipe Transaction | -7.27612E+17 | Monterey Park | CA | 91754 | 5411 | | No |
| 0 | 0 | 2002 | 9 | 2 | 6:22 | \$120.34 | Swipe Transaction | -7.27612E+17 | Monterey Park | CA | 91754 | 5411 | | No |
| 0 | 0 | 2002 | 9 | 2 | 17:45 | \$128.95 | Swipe Transaction | 3.41453E+18 | Monterey Park | CA | 91754 | 5651 | | No |
| 0 | 0 | 2002 | 9 | 3 | 6:23 | \$104.71 | Swipe Transaction | 5.81722E+18 | La Verne | CA | 91750 | 5912 | | No |
| 0 | 0 | 2002 | 9 | 3 | 13:53 | \$86.19 | Swipe Transaction | -7.14667E+18 | Monterey Park | CA | 91755 | 5970 | | No |
| 0 | 0 | 2002 | 9 | 4 | 5:51 | \$93.84 | Swipe Transaction | -7.27612E+17 | Monterey Park | CA | 91754 | 5411 | | No |
| 0 | 0 | 2002 | 9 | 4 | 6:09 | \$123.50 | Swipe Transaction | -7.27612E+17 | Monterey Park | CA | 91754 | 5411 | | No |
| 0 | 0 | 2002 | 9 | 5 | 6:14 | \$61.72 | Swipe Transaction | -7.27612E+17 | Monterey Park | CA | 91754 | 5411 | | No |

Figure 3-13 Sample credit card transaction data

The GitHub repository provides a Jupyter Notebook for training a model and generating the following ONNX model:

`ccf_220_keras_lstm_static-0S.ipynb`

The Notebook starts with splitting the data into testing, training, and validation sets. Before splitting the data, the following process is done to have fair representation of each credit card in each set:

1. Find all the credit card numbers and put them in a list.
2. The six previous customer credit card transactions are grouped together.
3. Split the remaining transactions randomly into training, testing, and validation test sets such that:
 - a. 50% of the data is put in the training set.
 - b. 30% of the data is put in the validation set
 - c. 20% of the data is put in the test set.

Note: It is assumed the current transaction and the six previous transactions for a credit card provide sufficient context to identify whether a transaction is fraudulent.

Selecting the fraud detection model

Like the use case in 3.2, “Credit risk assessment use case scenario” on page 48, we need a DL neural network for creating a model that can be used for detecting fraudulent credit card payments.

This time, we deal with *time series data*, which means that we need the data for the current transaction and the six previous transactions for the credit card to decide whether the transaction is fraudulent.

For time series data sets, a long short-term memory (LSTM) model is an optimal option to learn the relationships between current and past transactions. In LSTM models, additional layers of neurons are used to retain the weights longer, which helps with accounting for the historical data when evaluating the input data.

In the fraud detection solution, an LSTM model is implemented by using the code in Example 3-6.

Example 3-6 Creating an LSTM model

```
lstm_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(units[0], input_shape=tf_input, batch_size=7,
time_major=True, return_sequences=True),
    tf.keras.layers.LSTM(units[1], return_sequences=True, time_major=True),
    tf.keras.layers.Dense(output_size, activation='sigmoid')
])
```

This LSTM model implementation results in the following model parameters:

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|---------------|--------------|---------|
| lstm (LSTM) | (7, 16, 200) | 336800 |
| lstm_1 (LSTM) | (7, 16, 200) | 320800 |
| dense (Dense) | (7, 16, 1) | 201 |

Total params: 657,801
 Trainable params: 657,801
 Non-trainable params: 0

Data mapping and training the fraud detection model

After the training, validation, and testing sets are created, the next step in the Notebook is to map the data into reasonable numerical values. We do this task so that DL algorithms can process the non-numeric data within our data set. We can convert some of the data by using data mappers from the scikit-learn Python package. Also, some defined custom mappers are used for converting the data into representative values. Custom mappers include the following items:

- ▶ timeEncoder for converting time to numerical representative values
- ▶ amtEncoder for converting the payment amount
- ▶ decimalEncoder for converting names such as city names or merchant names

After the data preparation and mapping completes, the TensorFlow “fit” method is used to train the model, and the model is validated. Sixteen batches of training inputs are sent to the model at once, and checkpoint data consisting of the model weights is saved during each training epoch to the directory that is referenced in `checkpoint_dir` (see Example 3-7). This checkpoint data is used later to restore the model.

Example 3-7 TensorFlow training parameters

```

steps_per_epoch = 50000
checkpoint_dir = "./checkpoints/ccf_220_keras_lstm_static/"
filepath = checkpoint_dir + "iter-{epoch:02d}/model.ckpt"
batch_size = 16

#callback to periodically save the model during training
cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=filepath,
save_weights_only=True, verbose=1)

#generate batches of training inputs
train_generate = gen_training_batch(tdf,mapper,train_indices,batch_size)

#train the lstm model
lstm_model.fit(train_generate, epochs=20, steps_per_epoch=steps_per_epoch,
verbose=1, callbacks=[cp_callback])

```

The accuracy of the model on the test data was found to be over 99%. However, accuracy is a misleading metric when dealing with a mis-balanced data set, where the positive class (fraud) is less than 1% of the data set. In our data set, a model might predict that all transactions are legitimate, and have an accuracy of 99.88%. It is more valuable to examine confusion matrices and F1 scores on test sets.

An example confusion matrix for our use case is shown in Table 3-1 and allows a holistic view of how our model predicts the legitimacy of transactions. The rows of the confusion matrix represent what our model predicted for the test data, either classifying as legitimate or fraudulent. The columns of the confusion matrix represent what the classification of the test data is. A confusion matrix can be used to visually examine how fraudulent and legitimate transactions might be getting misclassified.

Table 3-1 Labeled confusion matrix

| | Actually a legitimate transaction | Actually a fraudulent transaction |
|--|--|--|
| Predicted as a legitimate transaction | True Negative Count | False Negative Count |
| Predicted as a fraudulent transaction | False Positive Count | True Positive Count |

Our goal is to maximize the amount of True Negatives and True Positives (correct predictions) that our model is predicting. The metrics within the confusion matrix are described in further detail below:

- ▶ True Negatives: The number of legitimate transactions that are correctly classified as legitimate.
- ▶ False Negatives: The number of fraudulent transactions that are incorrectly classified as legitimate.
- ▶ False Positives: The number of legitimate transactions that are incorrectly classified as fraudulent.
- ▶ True Positives: The number of fraudulent transactions that are correctly classified as fraudulent.

In addition to examining confusion matrices, also explore the F1 Score metric. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. When the classes in an ML problem are mis-balanced, F1 scores are an effective way to determine the performance of the model.

The *precision* metric formula is shown below. *Precision* is the ability of the classifier *not* to predict intuitively a legitimate transaction as a fraudulent one.

$$\text{Precision} = \text{True Positive} / (\text{True Positive} + \text{False Positive})$$

The *recall* metric formula is shown below. *Recall* is the ability of the classifier to find intuitively all the fraudulent transaction in our test set.

$$\text{Recall} = \text{True Positive} / (\text{True Positive} + \text{False Negative})$$

From precision and recall, we can calculate the F1 Score, for which the formula is shown below:

$$F1 = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

The confusion matrices of the first and second test sets are shown in Table 3-2 and Table 3-3.

Table 3-2 Confusion matrix for the first test set

| | Actually a legitimate transaction | Actually a fraudulent transaction |
|---------------------------------------|-----------------------------------|-----------------------------------|
| Predicted as a legitimate transaction | 99849 | 33 |
| Predicted as a fraudulent transaction | 1 | 117 |

Table 3-3 Confusion matrix for the second test set

| | Actually a legitimate transaction | Actually a fraudulent transaction |
|---------------------------------------|-----------------------------------|-----------------------------------|
| Predicted as a legitimate transaction | 4862456 | 1595 |
| Predicted as a fraudulent transaction | 327 | 5622 |

The F1 scores for the two test sets and the commands to obtain those F1 scores are outlined in Example 3-8.

Example 3-8 F1 scores for the two test sets

```
#print F1 score for first test set
print(f1([[99849,33],[1,117]]))

#print F1 score for second test set
print(f1([[4862456,1595],[327,5622]]))

->0.873134328358209
->0.8540179249582257
```

We see that in our first test set that 33 fraudulent transactions were incorrectly classified as legitimate out of 150, but only one legitimate transaction was not classified correctly by the model. For our second test set, 1595 fraudulent transactions were incorrectly classified as legitimate out of 7217, and 327 legitimate transactions were incorrectly classified as fraudulent. The model achieved good F1 scores for both test data sets.

Now that the TensorFlow model is ready, the next step is to convert the model to the ONNX format, deploy the model to WMLz, and start serving it. ONNX conversion is described and demonstrated in “ONNX conversion” on page 67.

3.3.2 Fraud detection model deployment and evaluation on IBM zSystems

Now that the TensorFlow LSTM model is trained, we must decide how we deploy the credit card fraud model on IBM zSystems. Credit card transactions are high volume and demand low latency, with hundreds of thousands of transactions occurring on an IBM zSystems per second, typically requiring single-digit millisecond response times. As such, the performance of an inferencing call during a credit card transaction must also be high throughput and low latency. Because most credit card transactions are processed by using z/OS, a deployment option that has an affinity to z/OS applications should be leveraged. WMLz is a natural choice due to its ability to simply deploy models close to z/OS applications and specifically in CICS regions.

Importing and deploying the model with WMLz

WMLz does not support TensorFlow models, so we must convert the model to an ONNX format and deploy it by using WMLz. WMLz offers two options that can be leveraged during the deployment of a model on IBM z16 to increase inferencing performance: micro-batching, and on-chip AI acceleration.

When micro-batching is enabled during the deployment of an ONNX model in WMLz, the WMLz ONNX scoring engine queues a configurable quantity of inferencing requests before sending them to the IBM Integrated Accelerator for AI on z16. By processing multiple inferencing requests at once, the AI accelerator maximizes throughput and reduces the response latency compared to processing inferencing requests one at a time. In addition to the customizable micro-batching quantity, a maximum latency time can be defined for the deployment, which means that the scoring server waits only a certain amount of time before passing the accumulated inferencing requests to the Integrated Accelerator for AI. For a model to be eligible for micro-batching, it must have one input dimension that is dynamic and one output dimension that is dynamic.

To leverage the AI accelerator, the ONNX model must be compiled by using either the zDLC or the ONNX-Multi-Level Intermediate Representation (MLIR) compiler. As of IBM z16, the following operations are supported on the IBM Integrated Accelerator for AI:

- ▶ LSTM Activation
- ▶ GRU Activation
- ▶ Fused Matrix Multiply, Bias op
- ▶ Fused Matrix Multiply (with broadcast)
- ▶ Batch Normalization
- ▶ Fused Convolution, Bias Add, ReLU
- ▶ Max Pool 2D
- ▶ Average Pool 2D
- ▶ Softmax
- ▶ ReLU
- ▶ Tanh
- ▶ Sigmoid
- ▶ Add
- ▶ Subtract
- ▶ Multiply
- ▶ Divide
- ▶ Min
- ▶ Max
- ▶ Log

For the most up-to-date list of IBM Integrated Accelerator for AI supported operations, see [Leveraging the IBM z16 Integrated Accelerator for AI](#).

Because our credit card fraud model contains multiple LSTM layers, the AI accelerator is leveraged. During the deployment of our model, WMLz uses the zDLC, which is run in a container within a zCX instance to compile the model to target the IBM z16 Integrated Accelerator for AI.

Note: To use micro-batching or on-chip AI acceleration during the deployment of a model, the IBM zSystems Deep Neural Network (zDNN) library must be installed on the IBM z16. zDNN is the IBM open-source API library for using the Integrated Accelerator for AI. For more information, see [zDNN GitHub](#).

ONNX conversion

To convert the TensorFlow model to an ONNX format, we use the *tf2onnx Python package*.

We define variables that describe the model parameters; redefine the model that we created during training; and load model checkpoint weights into the newly defined model. As in 3.3.2, “Fraud detection model deployment and evaluation on IBM zSystems” on page 66, we define the number of units in the two LSTM layers of the model.

We define the input size of the model (204 features) and the input shape that the first layer of the TensorFlow model expects. There is one output from our model, which represents whether the transaction was fraudulent or legitimate.

We define the path to the directory where we saved model parameter values during the training of the model. We use the final checkpoint data that is recorded during the 20th epoch training. This process is outlined in Example 3-9.

Example 3-9 Redefining the original LSTM model variables

```
#define the number of units in the two LSTM layers
units = [200,200]

#define the input size and input shape of the model
input_size = 204
tf_input = ([None, input_size])

#define the output size
output_size = 1

#define the path to the directory where model checkpoint values from model
training are saved
restore_dir = "./checkpoints/iter-20/model.ckpt"
```

We redefine the model with the same layers and parameters that we used during the original creation and training of the model (see Example 3-10).

Example 3-10 Redefining the model with original layers and parameters

```
new_model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(units[0], input_shape=tf_input, batch_size=7,
        time_major=True, return_sequences=True),
    tf.keras.layers.LSTM(units[1], return_sequences=True, time_major=True),
    tf.keras.layers.Dense(output_size, activation='sigmoid',name='dense')
])
```

Because we defined our LSTM layers with the **time_major** parameter as True, the model input has the format (transaction series, batch, features), with the series data coming first.³

We might notice that the variable **batch_size** is representative of the transaction series length in our case. As shown in Example 3-11, the model is expecting seven transactions (six previous transactions and the current transaction), each with 204 features as input.

Example 3-11 LSTM model input shape

```
new_model.input_shape
(7, None, 204)
```

The None value in the second dimension of the input shape (see Example 3-12 on page 69) is representative of the dynamic batch size of the model. This dynamic dimension allowed us to use batching during model training to send multiple transaction sequences to the model at once. For inferencing of the model, having a dynamic dimension in the model input means that the model can inference several customer transaction series for fraud at the same time, increasing inferencing throughput.

³ https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM

We now load the checkpoint data from our restore directory as the weights of our new model and compile the model with the same optimizer algorithm and loss function that we used originally (see Example 3-12).

Example 3-12 Loading weights and compiling a new model

```
#load weights from checkpoint data in restore directory
new_model.load_weights(restore_dir)

#compile new model with original optimizer and loss function
new_model.compile(optimizer='adam', loss='binary_crossentropy')
```

With the TensorFlow model restored to a “trained” state, we are now ready to convert the model to an ONNX format (see Example 3-13).

It is important to specify explicitly the operation set (opset) version to be used during the conversion. At the time of writing, The WMLz ONNX engine supports Version 13 of ONNX opsets.

Example 3-13 Converting the model to the ONNX format

```
#Save the converted onnx model to file 'ccf.onnx' in current working directory
save_path = "ccf.onnx"

#convert model to ONNX format and set opset
onnx_model = tf2onnx.convert.from_keras(new_model,output_path=save_path,opset=13)
```

Reshaping an ONNX model to take advantage of WMLz micro-batching

Now that the model is in the ONNX format, we must explicitly define the input and output shape of the model to leverage the micro-batching functions of WMLz. To use micro-batching, WMLz requires one dynamic dimension in both the input and output of the model. As you can see in Example 3-14, we use the onnx Python package, which has a useful function for redefining the dimensions of the model input and output. You can find the package at this [GitHub repository](#).

Example 3-14 Reshaping the ONNX model for micro-batching in WMLz

```
#import onnx python package
import onnx

#define input and output shape to leverage micro-batching
from onnx.tools.update_model_dims import update_inputs_outputs_dims
```

The function that updates the onnx model dimensions requires the name of the input and output layers of the model. In Example 3-15, we store those layer names in the `input_name` and `output_name` variables.

Example 3-15 Assigning input and output layer names to variables

```
#store input and output layer names as variables
input_name = new_model.input_names[0]
output_name = new_model.output_names[0]
```

As shown in Example 3-16, we explicitly specify the dimensions of our model input and output layer in the Python Dictionaries `input_dims` and `output_dims`.

Example 3-16 Updating the input and output dimensions to include a dynamic dimension

```
#specify the dimensions of model output layer
output_dims = {
    output_name: [7,-1,1] #-1 represents a dynamic dimension
}

specify the dimensions of model input layer
input_dims = {
    input_name:[7,-1,204]
}

#reload model
onnx_model = onnx.load('ccf.onnx')

#update input and output dimensions of model
new_onnx_model = update_inputs_outputs_dims(onnx_model,input_dims,output_dims)

#save the reshaped model
onnx.save_model(new_onnx_model,'ccf_resaped.onnx')
```

Because we specified `return_sequences` as `True` when redefining the layers of the model (see Example 3-10 on page 68), the output that is returned by the model contains seven fraud predictions, one for each of the seven transactions. To determine whether the “current” transaction is fraudulent, we analyze only the last inferencing result of the seven results that are returned.

As input, the model expects a series of seven transactions containing 204 features. The `-1` value that is present in both the output and input dimension array is representative of a dynamic dimension.

We reload the `onnx` model that we created earlier with the `tf2onnx` package, update the input and output dimensions of the model by using the `update_inputs_outputs_dims` function, and save the reshaped model (see Example 3-16). With these two dynamic dimensions explicitly defined in both the input and output dimensions of our ONNX model, we are ready to leverage micro-batching in WMLz during scoring requests.

Importing and deploying the model with WMLz

Now that our original TensorFlow model is converted to the ONNX format and its input and output dimensions are updated to include a dynamic dimension, we are ready to leverage the IBM z16 Integrated Accelerator for AI and micro-batching by using WMLz.

We import the model into WMLz by using the following process (see Figure 3-14 on page 71):

1. Log in to WMLz.
2. When we are logged in to WMLz, we select the **Models** tab and click **Import model**.
3. Select **From file** and type in the Name of the model, `ccf_model`.
4. Select **ONNX** as the **Model type**.
5. Browse your system for the model `.onnx` file.
6. Click **Import** to upload the model to WMLz.

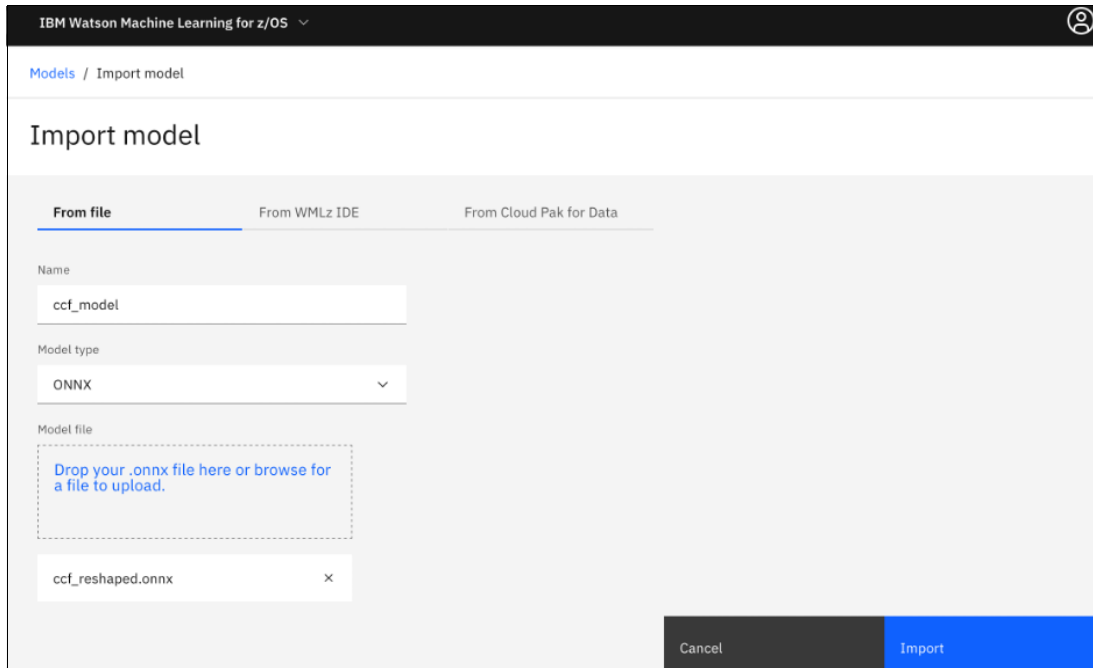


Figure 3-14 WMLz model import

After the model is imported into WMLz, it is compiled in the background by using the WMLz ONNX compiler, which is running in a container on a zCX instance. After the model has a “green check” as its Status indication on the **Models** tab, as shown in Figure 3-15, the model is ready to be deployed. To do so, click the three vertical-dot action menu, and select the **Deploy** option from the drop-down menu.

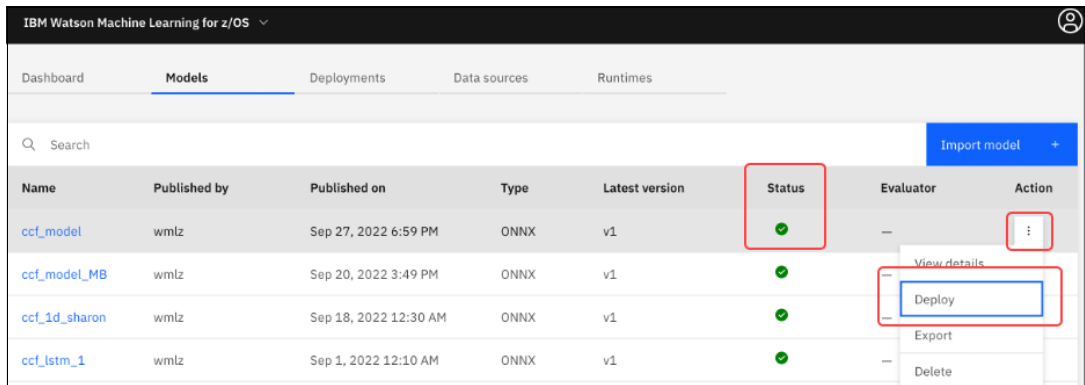


Figure 3-15 Deploying a model from the WMLz models section

Select **Models** → **Create deployment**. In the window that opens in Figure 3-16, complete the details to create a deployment environment for our model by completing the following steps:

1. Enter a unique name for the deployment in the Deployment namespace (ccf_deployment).
2. Select the deployment type **Online** from the drop-down menu.
3. Select the version of the model (**1**) from the **Model version** drop-down menu if multiple versions exist.

The screenshot shows the 'Create deployment' interface in IBM Watson Machine Learning for z/OS. The form is titled 'Create deployment' and contains the following fields and options:

- Model name:** ccf_model
- Deployment name:** ccf_deployment
- Deployment type:** Online (selected from a dropdown menu)
- Model version:** 1 (selected from a dropdown menu)
- Scoring service (standalone or cluster):** PLX7CIC1 (129.40.23.1:60083) (selected from a dropdown menu)
- Use on-chip AI accelerator
- Enable micro-batching
- Maximum batch size (maxBatchSize):** 8 (set via a spinner)
- Maximum latency in milliseconds:** 5 (set via a spinner)

At the bottom right of the form, there are two buttons: 'Cancel' and 'Create'.

Figure 3-16 Creating an ONNX deployment in WMLz

Decide what scoring server to which to deploy the model. In our example, we use a stand-alone scoring service that is running in a CICS region. This scoring service is embedded in a JVM, and we can use a CICS **LINK** command to perform online inferencing requests within a COBOL program.

Complete the following steps:

1. Select a Scoring service (stand-alone or in a cluster) from the drop-down menu:
PLX7CIC1 (129.40.23.1:60083)
2. Select the **Use on-chip AI accelerator** checkbox.
3. Select the **Enable micro-batching** checkbox:
 - a. Configure the Maximum batch size (maxBatchSize) by entering it or using the arrows to set the size limit (for our example, we set the limit to **8**).
 - b. Configure the Maximum latency in milliseconds by entering it or using the arrows to set the limit (for our example, we set the limit to **5**).

Note: As a best practice, tune these two server-side batching parameters to discover the combination that results in the best inferencing performance in your environment.

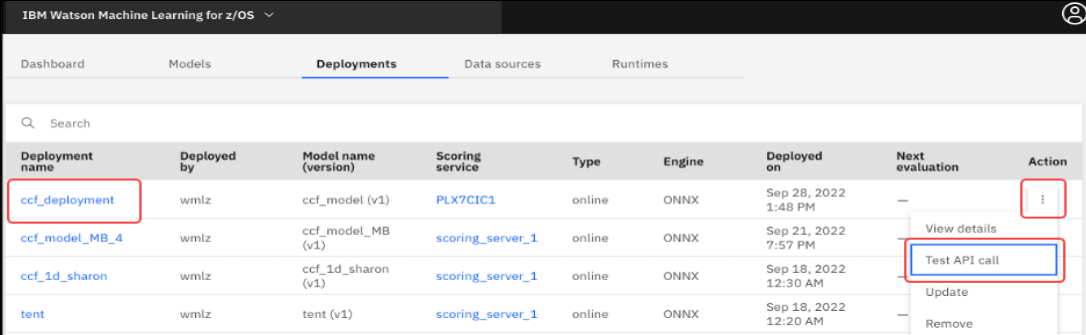
4. When you are satisfied with the deployment parameters, click **Create** to create the deployment.

During a performance test that uses the same model and WMLz deployment micro-batch size, we found that “An IBM z16 system can process up to 228 K z/OS CICS credit card transactions per second with 6 ms response time, each with an in-transaction fraud detection inference operation that uses a DL model.”⁴

Testing the model with WMLz

With our model deployed, we can now test an inferencing API call by completing the following steps:

1. Go to the **Deployments** tab of WMLz.
2. We find our deployment name, `ccf_deployment`, in the list, and we click the three vertical dots action menu to select the **Test API call** option from the drop-down menu, as shown in Figure 3-17.



| Deployment name | Deployed by | Model name (version) | Scoring service | Type | Engine | Deployed on | Next evaluation | Action |
|-----------------|-------------|----------------------|------------------|--------|--------|-----------------------|-----------------|---------------|
| ccf_deployment | wmlz | ccf_model (v1) | PLX7CIC1 | online | ONNX | Sep 28, 2022 1:48 PM | — | ⋮ |
| ccf_model_MB_4 | wmlz | ccf_model_MB (v1) | scoring_server_1 | online | ONNX | Sep 21, 2022 7:57 PM | — | View details |
| ccf_id_sharon | wmlz | ccf_id_sharon (v1) | scoring_server_1 | online | ONNX | Sep 18, 2022 12:30 AM | — | Test API call |
| tent | wmlz | tent (v1) | scoring_server_1 | online | ONNX | Sep 18, 2022 12:20 AM | — | Update |
| | | | | | | | | Remove |

Figure 3-17 WMLz Deployments window

On the Test API page, we can provide the values for an inferencing call in either a Table format (an array), or as a JSON variable. A sample JSON input is provided in the text file, `fraud-detection-sample-json.txt`, which you can obtain by following the instructions in Appendix B, “Additional material” on page 105. We chose to use the table input method.

⁴ Disclaimer: This 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 logical partition (LPAR) that was configured with six CPs and 256 GB of memory was used. Inferencing was done with WMLz 2.4 running on IBM WebSphere® Application Server Liberty 21.0.0.12 that used a synthetic credit card fraud detection model from [this GitHub repository](#), and the Integrated Accelerator for AI. Server-side batching was enabled on WMLz with a size of eight inference operations. The benchmark ran with 48 threads performing inference operations. The results represent a fully configured IBM z16 with 200 CPs and 40 TB storage. Results might vary.

As an input for this method of testing an API call, we can use a test vector from our earlier testing. Figure 3-18 shows the Input and Result after performing an inferencing call to the model by passing along the last seven transaction feature vectors as input and clicking **Submit**.

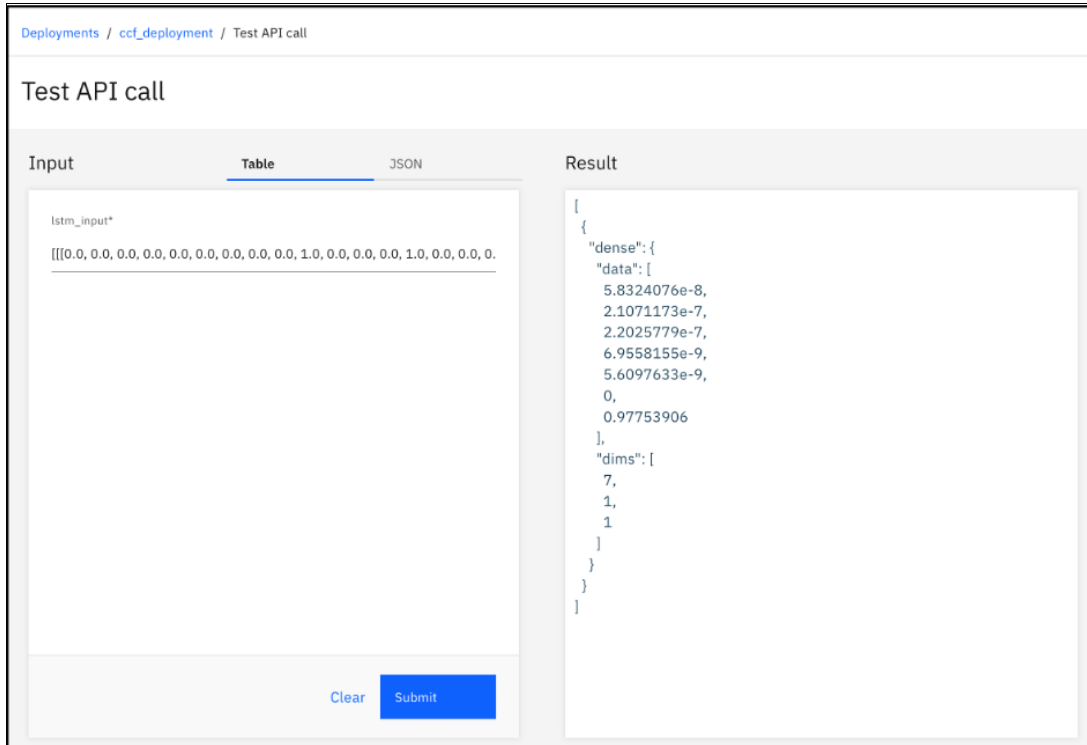


Figure 3-18 WMLz Test API call

In the Result pane, the model’s predictions are returned in JSON format. The key fields in the result include the following ones:

- ▶ dense: The name of the output layer of the model.
- ▶ data: The model’s prediction for each transaction.
- ▶ dims: The dimensions of the output that is returned by the model.

Looking at the last array value in the data key, we see the models prediction for the current credit card transaction. Our model performs a binary classification, which means that a probability score is returned with each prediction.

If the probability score of the model output is closer to 1 than 0, the transaction is considered fraudulent because we labeled fraudulent transactions as our positive class, or ‘1’, and legitimate transactions as the negative class, or ‘0’, when training the model.

Because the value 0.97753906 is closer to 1, the model predicted that the current customer credit card transaction is fraudulent. The first six customer transactions have probability scores that suggest that the transactions were legitimate.

After we confirmed that the model is predicting as expected, we can start integrating the model deployment’s REST API scoring services with z/OS applications.

By clicking the ccf_deployment text near the top of the browser, we can examine the specific details of the deployment, as shown in Figure 3-19 on page 75.

The screenshot shows the 'Deployments / ccf_deployment' page in IBM Watson Machine Learning for z/OS. It displays general information, API specification, and schemas for the deployment.

General

Name: ccf_deployment [Test API call](#)

Scoring endpoint: <https://129.40.23.1:60083/iml/v2/scoring/online/2ef340ba-0de9-41b2-8d51-1602668a4fe3>

Published by: wmlz

Deployed on: Sep 28, 2022 1:48 PM

Model name (version): [ccf_model v1](#)

Online feedback endpoint: —

Type: online

Engine: ONNX

Next evaluation: —

API specification: [Download](#) API specification for scoring endpoint.

Scoring service: [PIX7CIC1 \(129.40.23.1:60083\)](#)

Number of invocations: 52

Average elapsed time: 17.71 ms

Request header: See [API documentation](#) for details

Maximum batch size: 8

Maximum latency in milliseconds: 5

Schemas

Table | JSON | JSON schema

| Input schema | | | Output schema | | |
|--------------|-------------------|----------|---------------|-----------------|----------|
| Name | Type | Nullable | Name | Type | Nullable |
| lstm_input | float[7,None,204] | false | dense | float[7,None,1] | false |

Figure 3-19 WMLz deployment details

Here we can see specific details that are related to the API endpoint, including the address of the endpoint and a downloadable API specification that provides details to your applications about how they will use the API.

This API specification is in the format of a Swagger YAML file that details the operations that the API supports, the needed parameters, the usage of authorization tokens, the input format that is expected, and the output format that is returned. The Swagger file's machine-readable format is advantageous because it can be used to simplify the embedding of API calls within z/OS applications by using a service such as IBM Integration Bus or z/OS Connect EE.

From the Deployments details window of our ccf_deployment, we also see important performance metrics, such as the average elapsed time, as seen from the scoring server, for an inferencing call. The number of inferencing invocations is also displayed. We can confirm that the model input and output schemas are the same as the model input and output shapes that we defined in “Reshaping an ONNX model to take advantage of WMLz micro-batching” on page 69.

Our model is ready for consumption by CICS COBOL programs, either through RESTful API calls or by using a CICS LINK command.

3.3.3 Application integrations and considerations

When the model is deployed, there are two possible options in CICS to integrate WMLz and leverage the model for in-transaction AI inferencing:

- ▶ Use the CICS LINK command and the program ALNSCORE to invoke the model.
- ▶ Using REST API calls to invoke the model and get a decision.

Online scoring with ALNSCORE

WMLz provides a scoring service that is integrated in the CICS region through a program called ALNSCORE. It can be used in a CICS LINK command for scoring SparkML, Predictive Model Markup Language (PMML), and ONNX models by using special containers to transfer the scoring input and output between the COBOL program and ALNSCORE program, as shown in Figure 3-20.

After the CICS region is configured to run the scoring service, there are further steps that are involved in preparing the model, such as generating copybooks and Java helper classes that are based on the input and output schemas of the model, and by using special CICS containers to pass input parameters and receive the output from ALNSCORE.

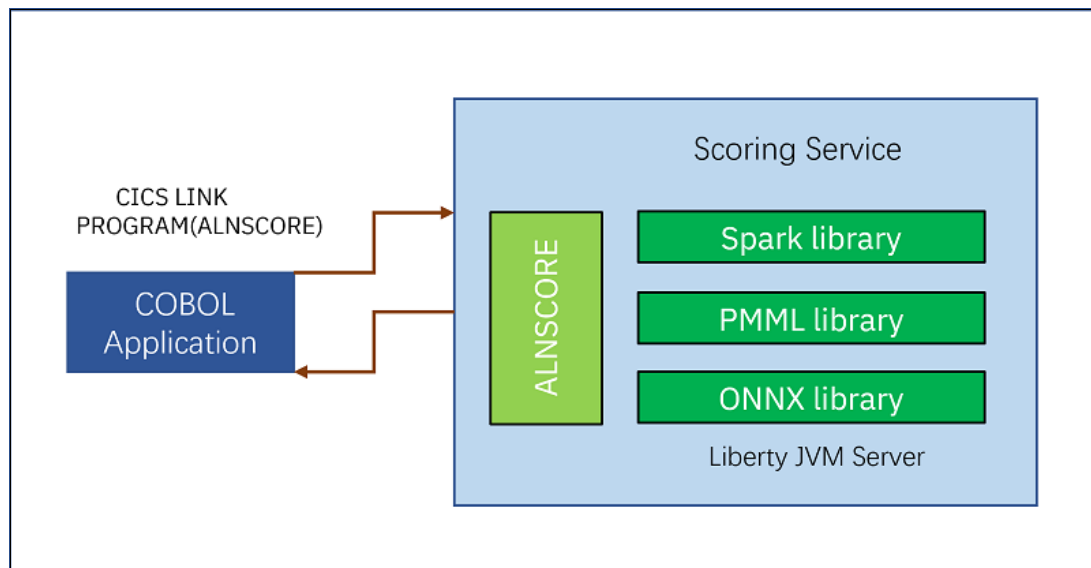


Figure 3-20 CICS integrated online scoring with ALNSCORE

This solution can yield optimized performance because the scoring server is colocated with the transaction within the CICS region itself. Coding the COBOL program, for example, by using the utilities that are provided in WMLz to construct the input/output layout for the scoring server might be easier compared to constructing the JSON input/output manually when using REST APIs.

For more information, see the following related WMLz documentation, including a sample CICS COBOL program and sample JCL/JOB to create the copybooks and Java helper classes:

- ▶ [Preparing a model for onlin scoring with CICS program ALNSCORE](#)
- ▶ [Configuring WML for z/OS scoring services in a CICS region](#)

Online scoring with a CICS program by using a REST API

Online scoring can also be implemented by making a REST API call to the scoring service on WMLz like what is shown in 3.2.3, “CICS application integration for real-time credit risk scoring” on page 58. The COBOL application that we used for our credit risk assessment use case, CREDRSCR.CBL, can be easily modified to include the sample JSON that is provided in the text file `fraud-detection-sample-json.txt`. For more information about obtaining both files, see Appendix B, “Additional material” on page 105.

To update the existing COBOL scoring application so that it can be used for the fraud detection model, complete the following steps:

1. Download the file CREDRSCR.CBL by following the instructions in Appendix B, “Additional material” on page 105, and open it in the editor of your choice.
2. In the WORKING-STORAGE SECTION of the DATA DIVISION section in the COBOL program, find the following line of code:

```
01 REC PIC X(100) VALUE SPACE.
```
3. Replace ‘X(100)’ with ‘X(8000)’ because the input is much longer than our previous input.
4. In the COBOL program, go to the READNEXT procedure in PROCEDURE DIVISION and update the procedure to read the JSON text file fraud-detection-sample-json.txt, as shown in Example 3-17.

Example 3-17 Updated READNEXT procedure to read the JSON sample file

```
READNEXT.  
    *Read one line from the file  
    EXEC CICS READNEXT FILE ('fraud-detection sample-json')  
        INTO (WS-STD-REC)  
        RIDFLD (WS-STD-REC-KEY)  
        RESP(WS-CICS-RESP)  
        RESP2(WS-CICS-RESP2)  
        RBA  
    END-EXEC.  
    IF WS-CICS-RESP = DFHRESP(ENDFILE) THEN  
        MOVE 'Y' to LASTREC  
    END-IF  
    MOVE WS-STD-REC TO REC.
```

5. Download the file DFH0WBC2.CBL by following the instructions in Appendix B, “Additional material” on page 105, and open it in the editor of your choice.
6. In the DFH0WBC2-00 procedure within DFH0WBC2-MAIN SECTION of PROCEDURE DIVISION in the COBOL program, update the token for the model by replacing the text with the most current token.
7. When creating the CICS resources by using the process that is outlined in CICS Application-CRED.docx, which you can find by following the instructions in Appendix B, “Additional material” on page 105, ensure that HOST (the host component of the URI) and PATH (the path component of the URI to which the map applies) attributes of the URIMAP matches the URL for the fraud detection model that is deployed to WMLz (scoring endpoint in Figure 3-19 on page 75).

Fraud detection solution

One example of leveraging WMLz for detecting fraudulent activities is through a payment system. From the fraud detection flow diagram in Figure 3-21, the customer initiates the following flow of actions:

1. Payee initiates a payment transaction to the system through the UI.
2. When the request is received, the fraud detection model is invoked for inferencing.
3. The model gathers the available historical transactions for this customer, compares them against the current payment that was made, and returns a prediction on whether the transaction is fraudulent.
4. Based on this prediction, an action is triggered to either reject or accept the payment, and the customer is notified.

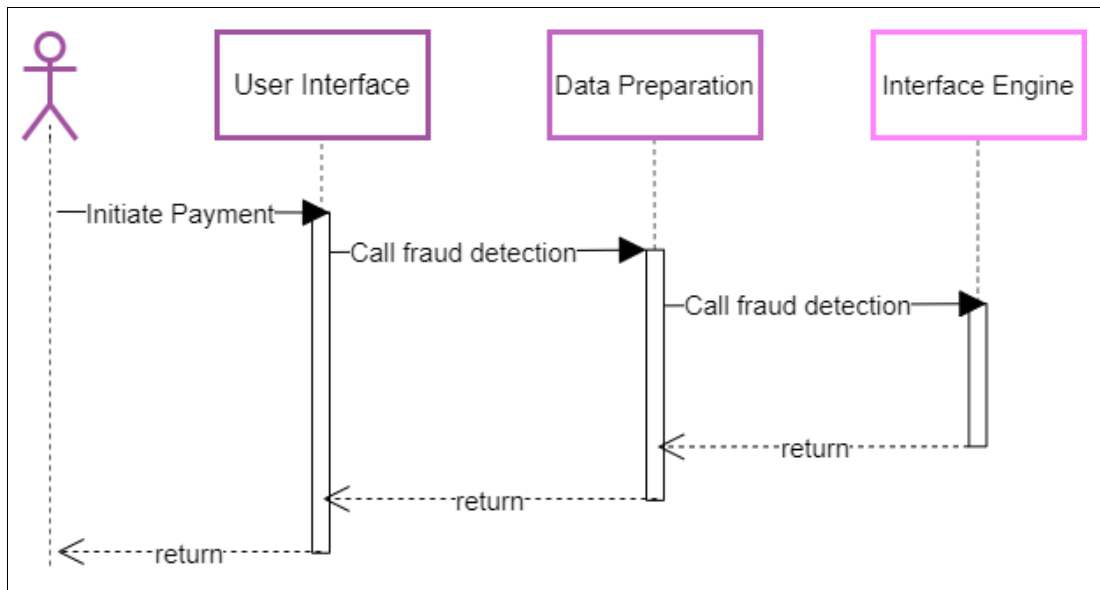


Figure 3-21 Fraud detection flow diagram

3.4 Summary

Organizations often miss the greatest opportunities that ML offers because tapping into those opportunities requires real-time predictive analytics.

To optimize large-scale business processes, which is a vital endeavor for your business, predictive analytics must occur right at the moment of every interaction. The good news is that you are likely already using the hardware to handle this endeavor; that is, the same system that is running your high-volume transactions (oftentimes IBM zSystems).

However, completing the effort often requires supportive leadership and a willingness for change throughout the organization. These high stakes include a tragic case of analysis paralysis and as a result, organizations often fumble the opportunity.

Do not let the profound capabilities of AI obscure the simplicity of its fundamental duty. The capability of learning from data to generate a predictive model translates into tangible value by way of the predictive scores the model generates, which in turn drives millions of operational decisions.

In some cases, unhurried, offline scoring works well enough. For example, in direct mail targeting, you might have 10 million contacts to be scored according to the likelihood of whether they are inclined to buy if sent a brochure. Even with that many records, your system must score only a few hundred per second to complete the task overnight, or only a couple of dozen per second if you have a few days and a few computers available.

For large-scale business processes, scoring cannot violate two main performance requirements:

- ▶ Time per transaction, which often requires performance times within single-digit milliseconds
- ▶ Transactions per second, which often requires the completion of thousands (even tens of thousands) of transactions per second

For some operations, the first requirement is more demanding than the second, which is the case for high-velocity operations. For example, an automated financial trading system that loses only a few milliseconds can lose the opportunity to purchase at the optimal price.

For other operations, the second requirement is a higher priority, which is the case for high-volume operations. For example, credit card fraud detection demands many transactions be processed per second for the sheer overall volume of transactions that are occurring.

To achieve predictive analytics within large-scale operations without violating performance requirements, you can perform model training on another system. Because this first phase of ML training is almost always performed offline, real-time systems that are busy handling operations do not need to be burdened.

Instead, this “heavy-lifting” number crunching process is best run with separate resources that are allocated to data scientists for model development. Also, you can use your operational system to perform scoring with the model.

To meet performance requirements, including those requirements that are dictated by SLAs, your organization should follow best practices by using a high-performance system such as IBM zSystems that operates on-premises rather than in the cloud. This system can likely handle the process, or in the case of IBM zSystems handles the introduction of model scoring, and integrate it so that only a limited impact on its overall performance.

Deploying predictive models in real time seizes the greatest of opportunities of ML. The highest-frequency operations are likely to be the most abundant because velocity means volume.



Other use case scenarios

Driving smart evolution within your enterprise can be fully realized with the help of artificial intelligence (AI) services. By using and accelerating AI integration throughout your enterprise, you can generate better business growth and outcomes in many areas within the business.

Whether it is AI for IT operations, advertising, health care, financial operations, risk and compliance, or security, AI solutions provide cross industry capabilities. You can improve customer service by cutting costs and providing faster, better service.

You also can realize an increase in workflow efficiency, reduce churn, and manage applications and infrastructure across environments. Supply chain insights can be garnered in real time, which enable agile operations control, or helps halt financial crimes, such as money laundering, as they occur, for improved regulatory compliance.

Whichever use case best aligns with your business challenges, an AI solution is available that fits your needs to solve your problems.

This chapter includes the following topics:

- ▶ 4.1, “The overall picture of your AI use case” on page 82
- ▶ 4.2, “Detecting CICS anomalies on IBM zSystems scenario” on page 89
- ▶ 4.3, “Object detection from geospatial images for cadastral analysis on IBM zSystems” on page 92

4.1 The overall picture of your AI use case

When looking at possible AI use cases on IBM zSystems, start with identifying which business problems that you want to address by using AI technologies. After you understand the business challenges that need addressing, for example, by participating in an AI on IBM zSystems Discovery Workshop, you can match that information with one of the more generalized use cases that are defined by IBM.

After you define the business challenges and select a use case, you can start discovering what AI technologies and open source frameworks that are available to integrate into your IBM solution and enable your goals.

The next step is to explore and choose an enterprise solution for your AI implementation on IBM zSystems. After you have all of that information and decisions are made, you are ready to start integrating the solution into your mission-critical business applications, and begin to fully realize the benefits achieved.

For example, after potential use cases are identified, you can explore available AI technologies and frameworks that were introduced in “Why AI on IBM zSystems” on page 6. When the evaluation is done, ask yourself the following questions:

- ▶ Where does the data that is used for training originate from and how is it stored?
- ▶ How large is the training data and how frequently is it updated?
- ▶ From where does the data that was used for the AI training and inferencing originate?
- ▶ Where does the application that uses the AI model run?
- ▶ What are the latency and data security requirements for the application that you are trying to infuse AI into?

The answers to these questions reveal whether the AI use case has good affinity for IBM zSystems and must be implemented with or on IBM zSystems.

Next, we describe the typical AI on IBM zSystems use cases that are seen by IBM. At a high level, these use cases can be categorized into two categories: business-related and IT-related.

There are many more potential use cases than the ones that are mentioned here. Because AI use cases are highly individualistic, a use case discovery workshop like the AI on IBM zSystems Discovery Workshop that we describe in 5.2, “What is next for your AI on IBM zSystems journey” on page 96 is highly recommended.

4.1.1 Business-related use case scenarios

The main reason to run AI on IBM zSystems for business use cases is to infuse AI into every business transaction and decision that is running on the IBM zSystems platform, and to derive real-time insights from predictive analytics to guide better, more informed business decisions.

Deep learning (DL) can be used for some traditional business problems, such as fraud detection and credit risk scoring. Consider the use of DL when you have a large amount of training data because DL is powered by data. The more data that is fed to the algorithm, the better the performance of the trained model.

For more information about a DL business-related use case scenario, see 4.3, “Object detection from geospatial images for cadastral analysis on IBM zSystems” on page 92.

Fraud detection

One example of a business-related use case is fraud detection. Fraud detection is of vital importance for banks, insurance agencies, and other organizations that deal with massive amounts of financial transactions and are at higher risk of suffering from financial fraud.

Traditionally, fraud detection typically uses a rule-based engine approach, which has the following two main disadvantages:

- ▶ Detection occurs after the transaction completes in a “pay and chase” mode. The losses that are caused by these fraudulent transactions are difficult to recover because the transactions already are complete.
- ▶ Rule-based engine approaches cannot adapt quickly for new fraud patterns, which lead to outdated detection methods and undetected fraudulent behavior.

However, AI methods are useful when there is a need to learn new fraud patterns quickly. AI methods can enrich the existing rule-based engine approach to fraud detection by increasing the accuracy of models through reduced false positive and false negatives. The goal here is not to remove the rules-based approach to fraud detection, but instead enrich it with AI for better and more accurate results.

False positives create more overhead through process bottlenecks because they must check every false positive by using a manual process and negatively affect revenue due to customer friction when valid transactions are rejected. False negatives are fraudulent transactions that slip through the cracks and incorrectly labeled as a valid transaction, which results in loss due to fraud (sometimes even permanent financial losses).

According to a Celent study that was commissioned by IBM, applying DL models for anti-fraud efforts resulted in a 36% decline in false positives. Additionally, the significant number of false positives that is experienced in anti-money laundering (AML) operations, which led to severe operational burden and costly regulatory fines, can benefit from leveraging DL models to help improve the accuracy of AML behavior detection and reduce false positives.¹

By using the IBM Watson Machine Learning for z/OS (WMLz) enterprise solution, you can continuously monitor the model and retrain it to adapt to new fraud patterns automatically. Moreover, the high performance online inferencing capability of IBM zSystems enables you to embed the fraud detection flow as part of the online transaction to detect a fraudulent transaction *before* it completes. Allowing your organization to be proactive instead of reactive in their fraud detection efforts mitigates loss and recovery costs.

Supervised and unsupervised machine learning (ML) approaches can be used for fraud detection modeling. With supervised learning, you can create classification models that use labeled data, for example, fraud versus non-fraud.

One key challenge of supervised ML modeling for fraud detection use cases is the lack of balanced training data because classification models expect balanced data. You can mitigate this challenge by using techniques such as data sampling and grouped models.

If you use unlabeled data for fraud detection modeling, you can leverage unsupervised learning, which is commonly used to create anomaly detection models. The lack of balanced training data is not a challenge for anomaly detection models because the models expect the data to be imbalanced.

¹ <https://www.ibm.com/downloads/cas/D0XY3Q94>

Implementation of an AI-powered fraud detection solution that can be embedded into the transaction and defines the business logic that deals with detection provides an important tool for detecting and preventing fraudulent transactions. For more information about the technical details regarding a fraud detection use case, see 3.3, “Credit card payments fraud detection use case scenario” on page 60.

Churn prevention

AI and ML technologies can be used when predicting the probability of churn and when to act before a customer is lost. You can apply AI churn prevention solutions to several industries, such as banking, insurance, retail, and health care.

If the critical customer-related data for churn analysis originates on IBM zSystems, it is optimal to perform model training and scoring on the IBM zSystems platform for the most benefits. Training on the platform helps mitigate security exposures by removing the need to off load data from the IBM zSystems platform. It also helps avoid the cost of data masking, and ensures the most efficient churn prevention process. An alternative to training the model on IBM zSystems is to use pseudonymous customer data for training off-platform and then perform inferencing on IBM zSystems to still benefit from the platform’s inferencing optimizations.

Risk scoring

Taking from the concept of churn prevention, the attempt to score, or inference, by use of linear regression methods can be used for other use cases to predict any type of risk. For example, to predict the risk of credit card insolvency or fraud at the conclusion of a contract.

Implementing such a use case on IBM zSystems where your business-critical data is stored enables you to avoid data movement, leverage the colocation of data and applications and inferencing, and benefit from the proven value of the IBM zSystems infrastructure.

For more information about the implementation of a risk scoring use case, see 3.2, “Credit risk assessment use case scenario” on page 48.

Customer segmentation and insight

Through customer segmentation, you can gain AI-driven insight into your customer base and use that insight to personalize service and product offerings for single customers. This personalization and customer behavior prediction can be made at the ideal point in the transaction for optimal up- and cross-selling strategies.

Supply chain optimization

AI on IBM zSystems technology may be used for supply chain optimization or other optimization challenges.

Pre-built IBM solutions

IBM is using AI technology to infuse their own product with new capabilities. A short example for IBM Db2 for z/OS is described in “IBM Db2 13 SQL Data Insights”.

IBM Db2 13 SQL Data Insights

IBM Db2 13 for z/OS SQL Data Insights (SQL DI) is a newly introduced function that enables AI queries on existing Db2 for z/OS data by using built-in Structured Query Language (SQL) functions. It enables business users and application developers to create AI queries without offloading sensitive Db2 for z/OS data. It also lowers the barrier of using that data for AI by removing the requirement of data science skills.

Db2 13 for z/OS function level (FL) 500 offers three built-in AI functions, which are outlined in Table 4-1.

Table 4-1 SQL Data Insights semantic query types

| Db2 built-in AI semantic function | Semantic query type | Explanation of AI semantic function |
|-----------------------------------|---|---|
| AI_SIMILARITY | Similarity and dissimilarity query | Given the attributes of an object, find other objects with similar or dissimilar attributes. The result is a similarity or dissimilarity score -1.0 - 1.0, where -1.0 means that the values are least similar, and 1.0 means that they are most similar or dissimilar. |
| AI_SEMANTIC_CLUSTER | Inductive reasoning semantic clustering query | Given a cluster of up to three entities, calculate a semantic clustering score indicating how well other entities fit into that cluster. The result is a score -1.0 - 1.0, where a larger positive value indicates a better fit into the cluster than a lower value. |
| AI_ANALOGY | Inductive reasoning analogy query | Given a pair of related values, find other values that have the same relationship (A is to B as C is to ?). The result is an analogy score, where larger-valued positive results indicate a better analogy than smaller results. |

For more information about more architecture descriptions and example use cases, see Chapter 6, “SQL Data Insights”, in *IBM Db2 13 for z/OS and More*, SG24-8527.

4.1.2 IT-related use case scenarios

The emerging requirements in the business domain added increasing challenges to IT infrastructure management because the traditional IT operations techniques can no longer cope with the business transformation and agility needs.

Organizations are looking at how to integrate AI and ML in their IT operations to improve efficiency. Such integration enables quick resolutions and reduced monitoring so that IT teams can focus on what really matters. This integration is known as *IT Operational Analytics* (ITOA).

As the general system that is hosting the infrastructure for enterprise applications and transactions, large amounts of operational and performance data is generated on the IBM zSystems platform daily. Applying ITOA and ML capabilities on this data with IBM zSystems can provide the following key benefits:

- ▶ Enable faster and more accurate anomaly detection with real-time performance data collection in run time.
- ▶ Avoid the cost and security vulnerability of offloading operational and performance data off the IBM zSystems platform.

- ▶ Ensure that the latest data is available and accessible for modeling so that the system behavioral pattern that is derived is current.
- ▶ Use the ML solutions that are available on the IBM zSystems platform for easy adoption and implementation.

ITOA use cases on IBM zSystems are described in “Anomaly detection”.

Anomaly detection

Anomaly detection refers to the case of identifying abnormal behaviors of the system (or applications) by using ML techniques. Anomaly detection can be helpful and applicable to almost all the key components on IBM zSystems; for example, Db2 for z/OS system and applications, IBM Customer Information Control System (IBM CICS) transactions, IMS transactions, IBM MQ, and batch jobs.

All of these components generate large amounts of performance data, such as System Management Facility (SMF) data and logs, in their daily operations. SMF collects and records system and job-related information that can be used to generate different types of reports, such as profiling system resource use reports, and reliability reports. The key idea here is to use ML methodologies to build models to depict the normal behavioral pattern, or baselines of the key performance indicators (KPIs), which can then be used to score against new data to detect anomalies.

Compared with traditional anomaly detection methods that typically include the use of predefined thresholds for KPIs in monitoring tools or having domain experts analyze performance data manually, the ML approach supports detecting anomalies for a broader scope of KPIs with improvements in accuracy and efficiency while adapting automatically to changing workload patterns. In particular, unsupervised methods are suitable for this type of anomaly detection that is based on the time series approach.

Time series data is a sequence of data points that are collected over time. Each data point can have two metrics: the time when the data point was generated, and the value of the data point.

The KPIs that fit best for time series analysis are the KPIs that are time-oriented and repeatedly sampled; for example, the transaction volume, CPU time, and elapsed time. Most of the performance KPIs that are collected by the z/OS system, Db2, CICS, and IMS are time series data.

Combined with a statistical profiling approach, you can build baseline models that define the normal upper bound and lower bound of the KPIs over the time where the baseline models can be used to score against the new data. Any data point that falls beyond the normal range of the KPI can be detected as an anomaly.

The time series method can discover the overall common trend of a certain KPI and the cyclic pattern of the KPI value. Therefore, the baseline models that are built carry the dynamic thresholds that can show the normal behavior of the KPIs across different days of the week, different hours of the day, or even on a minute interval.

Compared with the use of fixed thresholds, this ML method can detect anomalies more accurately and proactively. Figure 4-1 on page 87 shows an example of the use of a time series-based dynamic baseline model to detect anomalies for Db2 statistics trace, KPI ACTIVE_DBATS.

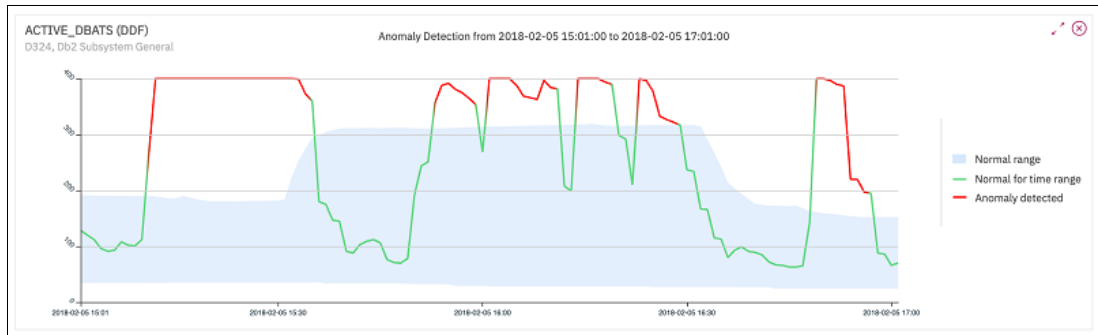


Figure 4-1 Using a dynamic baseline model for Db2 anomaly detection

For more information about an anomaly detection use case, see 4.2, “Detecting CICS anomalies on IBM zSystems scenario” on page 89.

Batch job management

Batch job management is one of the daily responsibilities of system operators and administrators, and is yet another area that can be optimized by AI and ML.

Batch job profiling

Batch job profiling is the recommended first step in batch job management and is a process that can help you better understand your batch jobs.

You might have hundreds, or even thousands, of batch jobs that are running on the system every day and belong to different batch applications. All of these jobs generate operational data that is collected as type 30 SMF records. By applying statistical and ML methodologies to the historical SMF 30 data records by way of visualization, time series analysis, correlation analysis, or another method, you can gain valuable insights regarding the execution pattern of the batch jobs.

These valuable insights can help you answer critical questions about job execution, including the following examples:

- ▶ How frequently are the jobs run: daily, weekly, or monthly?
- ▶ What are the most resource-intensive jobs on the system?
- ▶ What is the batch elapsed time for different nightly batch applications? Does a seasonal pattern exist?
- ▶ What are the influential jobs that affect the batch elapsed time?

Correlation analysis

A large percentage of the batch jobs, especially nightly batch jobs, connect to daily business transactions; for example, credit card payment batch processing. The credit card transactions are captured and accumulate during the daytime and wait to be settled by the nightly batch jobs.

In this case, some business factors (transaction volumes of the day, number of accounts, and so on) can influence the batch window at night. Combing those business factors with SMF 30 data records for correlation analysis, you can identify and visualize the relationship among online business transactions and batch applications, which creates the foundation for the next step of batch job management; that is, batch window prediction.

Batch window prediction

One key challenge of batch job management is to ensure that the batch window is respected, while simultaneously ensuring it does not overuse system resources. You can create ML models to predict the batch window to mitigate this challenge. The model can be created by using the data that is collected for the daytime transactions to form predictions of the batch window at night.

However, your batch window must show strong correlation to the daily business transactions. The prediction can be run nightly before the batch job starts so that you can evaluate whether you must allocate more system resources to enforce the allowed batch window.

The predictions also can be made spontaneously during the batch execution as an alternative approach. This feature is helpful for long-running batch jobs that can take hours to complete.

The general process flow is shown in Figure 4-2.

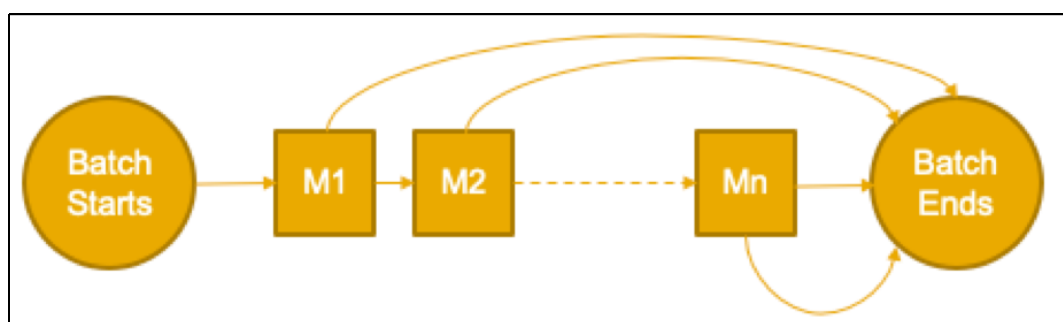


Figure 4-2 Batch job prediction during batch execution

In between the start of the first job of the batch through the completion of the last job of the batch, many small, medium, and large jobs are run in sequence or in parallel. How long the entire batch takes often is affected by some of the key jobs in the batch.

Through the combination of ML methods and experience from domain experts, you can identify a set of key jobs that are called *milestone jobs*. These milestone jobs are represented by M1, M2, ..., Mn in Figure 4-2.

After you identify the milestone jobs, a model can be built for each milestone by using the performance data that is collected by SMF 30 records for the jobs. This SMF 30 record includes data, such as, CPU time, elapsed time, and I/O counts.

During the batch execution, scoring can be called at every milestone to get a fresh prediction regarding when the batch finishes. In this way, you can monitor the batch evolution as it runs and dynamically adjust system resources to get the most optimal balance between batch window and resource consumption.

“Black-box” solutions

In the ITOA space, you also can use the “black-box” solutions that are provided by IBM to accelerate ML adoption in your company. These solutions use the capabilities of ML that you can benefit from without the cost of any data science efforts that often are needed for model development.

The “black-box” solutions on IBM zSystems offer the following key features:

- ▶ IBM Db2 AI for z/OS (Db2ZAI): Db2ZAI learns patterns from your unique operating environment to improve Db2 for z/OS application and system performance by using ML, with a focus on the following areas:
 - SQL optimization: Uses ML to help the Db2 Optimizer choose the best access path.
 - System assessment: Uses AI to learn what is normal, performs exception-based analysis, and provides performance tuning recommendations on the system level.
 - Performance Insights: Uses ML for interactive Db2 performance analysis on an application level.
 - Distributed connection control: Uses AI to learn the distributed connection behavior to control the handling of connections and threads.

For more information and resources regarding Db2ZAI, see “IBM Db2 AI for z/OS 1.5.0” on page 102.

- ▶ IBM zSystems Operations Analytics (IZOA): IZOA uses ML for anomaly detection in Db2 for z/OS and CICS. Through visualizations, it helps you find hidden problems in the workloads and perform root cause analysis quickly.

For more information and resources regarding IZOA, see “IBM zSystems Operations Analytics” on page 104.

4.2 Detecting CICS anomalies on IBM zSystems scenario

In this section, we describe an ITOA on IBM zSystems use case scenario in which a fictional customer, Customer X, uses WMLz with a focus on detecting CICS anomalies.

4.2.1 Background and business goal

To ensure the best performance and control resource consumption, Customer X implemented a performance reporting process for CICS and Db2 for z/OS. This performance reporting process runs regularly to collect, aggregate, correlate, and externalize CICS performance data from SMF records to CSV files.

The customer used an Analytics Engine with statistical algorithms to analyze the CSV files in addition to visualizing the data. Then, experts reviewed the data regularly to identify potential anomalies within the previous day.

The identification of the root cause was found through further manual analysis. With this approach, Customer X detected a specific number of anomalies in a specified period. However, this approach was still overly time-consuming and resource-intensive. Customer X needed a solution that detects anomalies quickly, and free up their experts’ time for more pressing matters.

The goal became to implement CICS performance anomaly detection by using WMLz. The implementation included the target to detect anomalies faster than Customer X was performing at the time. Cases also exist in which some anomalies were not detected easily by humans or static thresholds. This issue might be the case for a slow decrease or increase of a performance-related metric or an increase of a performance-related metric in a valley.

Figure 4-3 shows example scenarios that must be covered by the anomaly detection model. Trends in the consumption of service units (SUs) by a single transaction can be seen. Service units were chosen as an indicator and the anomaly detection is done on the transaction level.

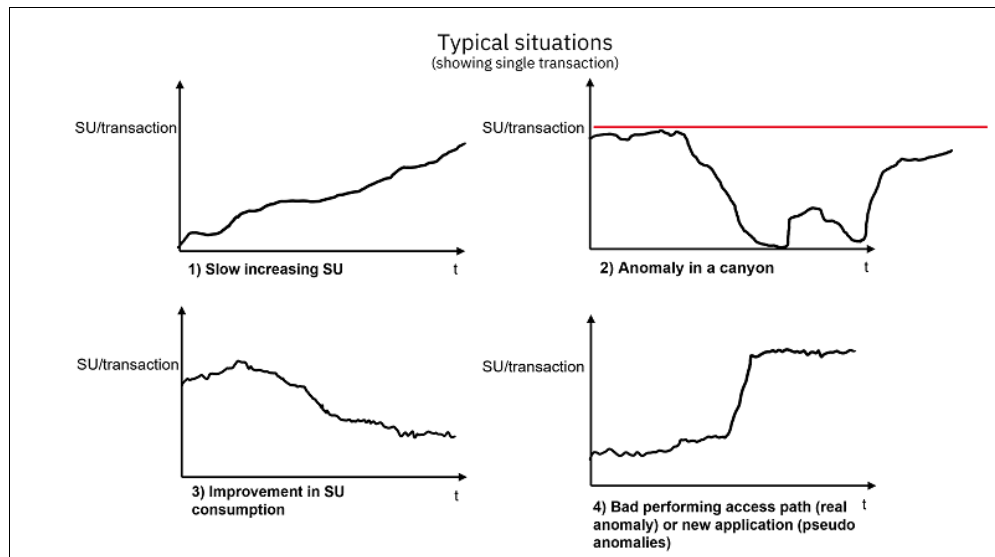


Figure 4-3 Examples of typical scenarios that are covered by the anomaly detection approach

The benefits of implementing such a solution for Customer X's use case are an increase in the speed of accurate anomaly detection. Especially after an application change, the quick detection of performance-related metrics is critical. As the AI solution, WMLz can detect more anomalies in a specific period than the initial manual approach of Customer X.

4.2.2 Architecture

This use case uses CICS performance-related SMF data that is converted to CSV format, which serves as the data source for WMLz and the AI model.

Thinking further, Customer X can use IBM Data Virtualization Manager (DVM) for z/OS to directly access the SMF data, which enabled an automatic data flow process and improves process speeds.

Figure 4-4 on page 91 shows the components and procedure flow of this use case architecture. The procedure includes the following steps:

1. SMF data is transformed to CSV data:
 - Serves as the data input for the model development and for scoring of the model to get the prediction.
 - Various timeframes were selected that consisted of weeks to months. For a production implementation, the usage of data that consists of years might be considered. However, the data scientist must consider that metrics can change with the introduction of a new application. They also must ensure that this issue is not classified as a false anomaly.
2. CSV data is loaded into the WMLz integrated development environment (IDE).
3. Data exploration is performed to understand the meaning of the provided data, identify correlations between metrics, and choose the relevant features for anomaly detection.
4. Dynamic baseline models for a selected subset of transactions are created and trained.

Having a trained model, the anomaly detection with WMLz base is performed by using the dynamic baseline models. For this process, we use Python in Jupyter Notebooks.

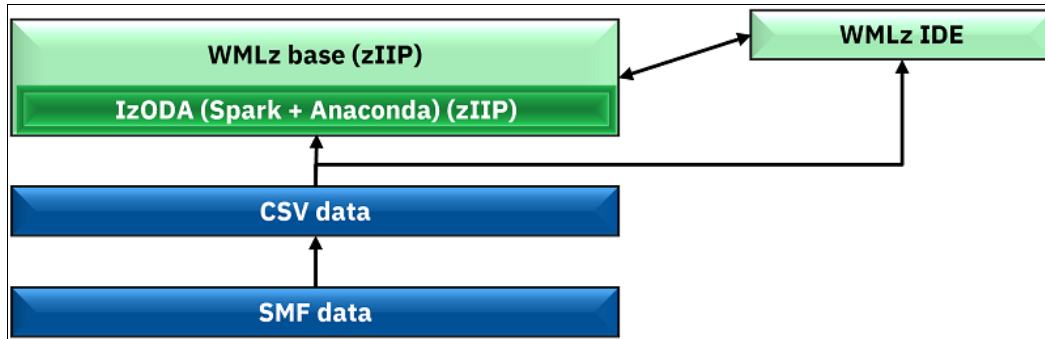


Figure 4-4 Components and data flow of our IT-based use case

Thinking further and establishing more AI on IBM zSystems use cases, IBM Cloud Pak for Data (CP4D) can be considered as a data science platform for creating and training the ML model instead of the WMLz IDE.

4.2.3 Summary

You can adapt this use case approach to other customers or use cases. We recommend the following steps (see Figure 4-5) to help you discover the value of AI on IBM zSystems.

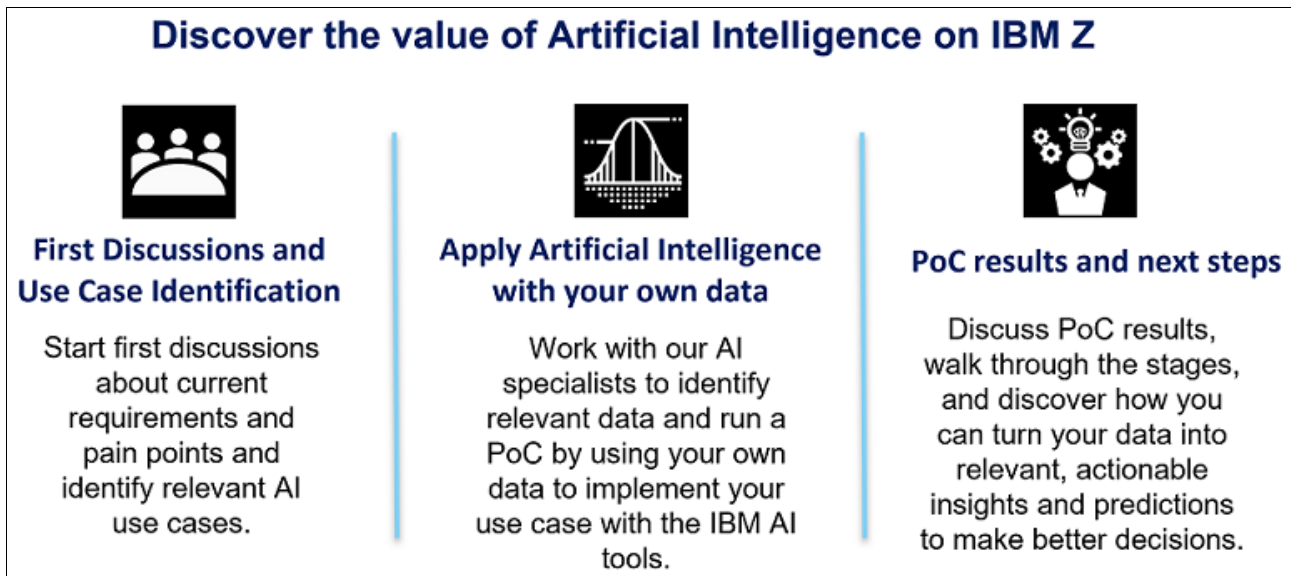


Figure 4-5 AI on IBM zSystems engagement actions

1. Facilitate a discussion with key stakeholders from multiple departments, such as IT, business, or data science departments, regarding your business goal and the selection of an AI on IBM zSystems use case.

As a best practice, include an AI on IBM zSystems Discovery Workshop, which is described in 5.2, “What is next for your AI on IBM zSystems journey” on page 96, as a part of this multi-disciplinary, facilitated discussion. Doing so helps all the teams that are involved learn about the solutions that are available for AI on IBM zSystems and identify use cases for your organization.

2. Identify relevant data for your use case and run a proof of concept (PoC) that is based on your data and environment.
3. Discuss and present the results of your PoC and begin discussing what your next steps must be to turn everything that you learned into actionable insights and predictions to enrich your business.

4.3 Object detection from geospatial images for cadastral analysis on IBM zSystems

In this section, we describe a business-related AI use case scenario on IBM zSystems in which a customer, Customer Y, use DL model inferencing on IBM zSystems for object detection in images.

4.3.1 Background and business goal

Being a trusted partner for various departments, Customer Y uses the IBM zSystems platform and related products. They want to use AI, particularly DL models, for geospatial images and cadastral data analysis, pattern recognition, and identification and comparison of buildings by using object detection.

To do so, they use the combination of their expertise and IBM expertise to create a PoC. The main objective of the PoC is to evaluate the Open Neural Network Exchange (ONNX) scoring capability on IBM zSystems that uses WMLz. The goal of the project is to verify that a DL model, created on any platform, can be taken and deployed on IBM zSystems, It provides the possibility to score directly on IBM zSystems for detecting buildings in the specific images.

As an IT service provider for public clients, Customer Y works to continuously improve their operational efficiency and qualities of service by using AI and ML capabilities. Because their core business is running on IBM zSystems, doing the model inferencing on IBM zSystems enables them to use IBM zSystems infrastructure for new types of workloads.

At the same time, Customer Y can benefit from the hardware consolidation, security, and high availability (HA) that is provided by the platform.

4.3.2 Architecture

Because the only objective was to verify the ONNX scoring capabilities on z/OS, IBM WMLz Online Scoring Community Edition (WMLz OSCE) was chosen.

WMLz OSCE simplifies the engagement and allows the focus to be on the technical evaluation. WMLz OSCE server is a Docker container that runs in IBM z/OS Container Extensions (zCX) instance on the z/OS logical partition (LPAR), as shown in Figure 4-6 on page 93. The container can be managed by the provided script that is included with the OSCE package. As a lightweight version of WMLz, the process of getting WMLz OSCE up and running can be done easily.

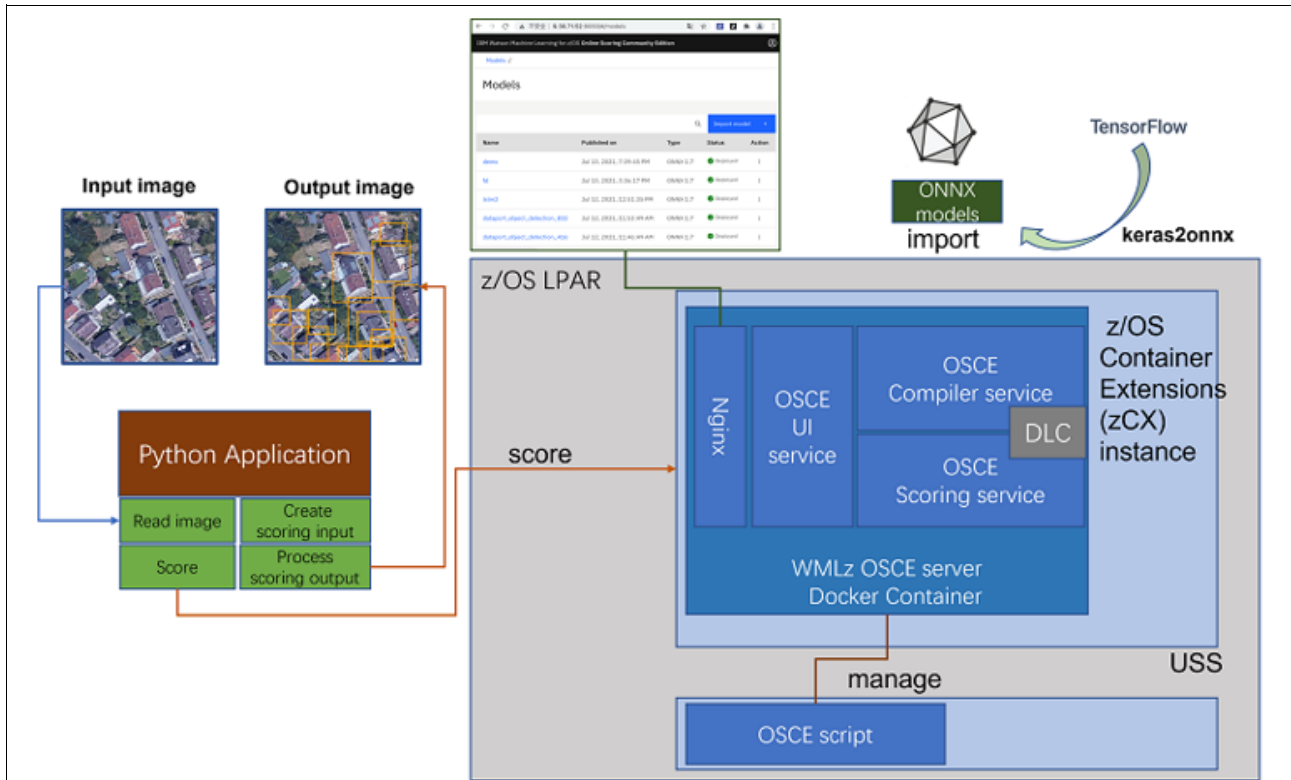


Figure 4-6 Object detection use case architecture

The WMLz OSCE server provides a web user interface from which the DL model, which is trained on any platform and converted into ONNX format, can be imported to the server in an intuitive and simple way.

After the model is imported, OSCE uses the IBM zSystems Deep Learning Compiler (zDLC) technology to compile the model into executable C++ code, and deploys the compiled model to the server. For each deployed model, a unique Representational State Transfer (REST) endpoint is exposed. Applications that are running on any platform can use the REST application programming interface (API) to call the model for online inferencing.

The model that is used for the project is an object detection “You Only Look Once” (YOLO) model that was trained by using the Keras high-level API of TensorFlow. YOLO is a convolutional neural network (CNN)-based algorithm that is used for detecting and recognizing objects in an image. YOLO is a popular model today because of its advantages in accuracy and efficiency.

After the model is trained, it is converted into ONNX format by using keras2onnx converter. The size of the converted ONNX model file is approximately 200 MB. The keras2onnx converter is available at this [GitHub web page](#).

The input of the model is a multi-dimensional array with type `float32[N,M1,M2,3]`. The first level “N” indicates the number of input images. The second and third levels “M1” and “M2” indicate the size of the input image, the height, and the width. The fourth level “3” carries the RGB value of each pixel in the image.

The model can take any number of images of any size for scoring.

The model features two outputs: bounding boxes and confidence score. Bounding boxes depict the rectangle boxes that enclose the detected objects in the image. The confidence score indicates how confident it is that a bound box encloses some object. In the PoC project, this ONNX model was imported to WMLz OSCE for testing the model online scoring capabilities.

4.3.3 End-to-end testing

In the PoC, aerial images with sizes 416 x 416 and 1664 x 1664 were used for the testing. A Python application was created to test building detection in the geospatial images end-to-end.

As shown in Figure 4-6 on page 93, the Python application can read in an image file from a local disk, transform the image to be represented by multi-dimensional array, and generate the scoring input. Then, it calls the WMLz OSCE server to use the imported YOLO model for inferencing given the scoring input.

After the scoring is done, the result is returned to the Python application, and is post-processed by the application to generate an output image with detected buildings enclosed by rectangle boxes.

4.3.4 Summary

The PoC was successful in proving that WMLz OSCE can run DL models on z/OS for detecting objects in images. One key challenge that was faced was two operators in the YOLO model were not supported by the zDLC, which was mitigated quickly by IBM.

Closely working with the IBM team is a valuable part of the journey to AI for Customer Y. The collaboration enables IBM to continuously improve and expand the list of supported ONNX operators. It also enables the customer to immediately receive the benefits from the supported operators.

Because of the success of the PoC, Customer Y is ready to take their solution to the next level of implementation and scale. As the next steps, Customer Y and IBM continue to collaborate and expand the solution for the use case. They move from the use of WMLz OSCE to the official version of WMLz, which can provide enterprise-grade model lifecycle management capabilities, HA, and more, for production deployment.

Significant performance improvements can be seen by model inferencing through Customer Y, who uses the IBM zSystems Integrated Accelerator for AI in z16, which is available as part of the Telum processor.



Key takeaways

This publication describes various methods that can be used to develop, train, and deploy artificial intelligence (AI) models on the IBM zSystems infrastructure, including integration points with other platforms and solutions. This chapter summarizes some of the key points that are addressed throughout the publication, and includes the following topics:

- ▶ 5.1, “Key technologies for AI on IBM zSystems” on page 96
- ▶ 5.2, “What is next for your AI on IBM zSystems journey” on page 96
- ▶ 5.3, “Future of AI technologies on IBM zSystems” on page 98

5.1 Key technologies for AI on IBM zSystems

Throughout this IBM Redpaper publication, we focused on key use case implementations that use IBM Watson Machine Learning for z/OS (WMLz) as a key component in an AI solution that leverages an AI model that was developed off-platform and then deployed to IBM zSystems by using WMLz. This configuration enables solutions that are vital to the IBM zSystems server, and enables enterprises to leverage the IBM zSystems platform strengths, for example, the IBM Integrated Accelerator for AI within the z16 Telum chip.

Combining the usage and integration of open-source technologies with the functions that are offered in WMLz supports the strategy of build and train anywhere, and then deploy and infuse on IBM zSystems. Following this strategy, you have access to many open-source container images that run on IBM zSystems, in addition to increased integration points of AI on IBM zSystems products.

Performing scoring and deployment on the IBM zSystems infrastructure is the best way to get the most out of your existing infrastructure investments and improve your inference performance metrics through optimized inferencing by using the Integrated Accelerator for AI, if it is present. This approach is the most practical and securable strategy due to the data gravity that the platform offers, which reduces or eliminates the need to move data between environments.

Combining the running of the AI model on the same Telum chip as the core transactions enables a more energy-efficient and much faster execution of the model when compared to using traditional accelerators, such as GPUs. A powerful GPU contributes to high energy consumption in the system. The Integrated Accelerator for AI can work much faster with the data on the cache structure of the Telum chip versus sending data to the GPU over the PCIe lanes to a GPU for inferencing.

From a customer's perspective, machine learning (ML) and deep learning (DL) open exciting new possibilities with timely, intelligent AI-driven insights. You can get started on your AI on IBM zSystems journey without any up-front cost with no-cost services, such as WMLz Online Scoring Community Edition (OSCE). Furthermore, the IBM continued focus on open-source frameworks allows for an easy transfer of skills and enables you to use the knowledge of your own data science team of experts.

5.2 What is next for your AI on IBM zSystems journey


Early in your AI on (and with) IBM zSystems journey, you should connect with your colleagues and determine about the benefits of AI on (and with) IBM zSystems. Also, you should cooperate with the data science team within your organization to combine knowledge and domain expertise because AI projects are a team effort.

A best practice approach is to engage in a no-charge, IBM Client Engineering AI on IBM zSystems Discovery Workshop, co-creating with an IBM squad of deeply skilled and multi-disciplinary technology experts. This workshop is designed to help you discover and better understand the AI technology on the IBM zSystems platform and develop the architecture, practices, and hands-on experience to successfully begin leveraging these AI and analytic capabilities to gain new insights from your business-critical workloads and data.

The workshop, which is outlined in Figure 5-1, brings together your business and technology context, sponsorship, subject matter experts (SMEs), data; and IBM global expertise, technical accelerators, and proven patterns to co-create innovative solutions for your business opportunities by leveraging the latest AI on IBM zSystems technologies, including the IBM z16 Integrated Accelerator for AI.


AI on IBM Z Discovery Workshop

IBM Client Engineering



Scope and content

- Presentation of the analytics and AI technologies and solutions on IBM Z and their place in your AI journey, including the new accelerator on IBM z16
- Advantages of performing analytics and inferencing directly on IBM Z
- Integration possibilities in existing development environments and business processes
- Joint development (Co-creation) of use cases for your environment through interactive hands-on experience in workshop discoveries on our infrastructure
- Streamline model deployment, development, and management
- Duration: approx. 1-2 days (dependent - scope and content is individually customizable)
- Format: virtual, in-person, or tailored to your needs
- Following: Implementation of a use case (outside the workshop)



Target groups

- Data Engineers/Scientists
- Strategists
- Developers
- Technical teams
- IT-/Enterprise-Architects
- Business leaders

Figure 5-1 AI on IBM zSystems Discovery Workshop

By engaging in this kind of a facilitated collaboration, you can accelerate your AI on (and with) IBM zSystems journey. At the end of this workshop, you achieve the following goals:

- ▶ Co-create a list of potential AI on (and with) IBM zSystems use cases for your environment.
- ▶ Learn best practice recommendations for using AI on the IBM z16 technologies that are described in the workshop.
- ▶ Provide skills enablement for your teams.
- ▶ Obtain recommendations for initial AI on IBM zSystems adoption.
- ▶ (Optional) Define use case requirements and outcomes, architecture, and a configuration and practical implementation plan.

If you are interested in this workshop, contact your local IBM zSystems Architect or your IBM zSystems Sales contact.

5.3 Future of AI technologies on IBM zSystems

The focus of this publication was heavily targeted on WMLz, which we used as the scoring and deployment platform for our implemented use case. Abid Alam, an IBM Product Manager for WMLz for AI on IBM zSystems, gave the following statement about the WMLz Strategy Direction:

“IBM WMLz enables organizations to operationalize ML and DL models on z/OS, where most transactions originate for IBM zSystems customers. This effectively extends their current investment in the platform for those models that require the lowest latency within a security-rich environment. WMLz offers a hybrid cloud approach to model training, deployment, and lifecycle management. Customers have the flexibility to train models anywhere they want and deploy those models on the IBM zSystems platform - closer and colocated with their mission-critical transactional workloads, which ensures scalability, security, and reliability while meeting the most stringent of service-level agreements (SLAs).

Supported models include Spark, Python (such as SnapML, XGBoost, and scikit-learn), Predictive Model Markup Language (PMML), and Open Neural Network Exchange (ONNX) DL models (such as TensorFlow, Keras, PyTorch, and Caffe). WMLz is the flagship end-to-end ML product and a key component in the overall AI on IBM zSystems strategy. In addition to business application use cases, it also enables solutions that are vital to the IBM zSystems system, including Db2 AI for z/OS, IBM zSystems Operations Analytics, and other upcoming AI-Powered solutions. The future direction of WMLz is focused on continuous product enhancements to support a wider array of use cases and the use of the Telum chip (an on-chip AI accelerator) to support DL models to accelerate the adoption of AI on IBM zSystems.”

Instead of WMLz, there are several other AI on IBM zSystems software solutions that are mentioned in this publication. Depending on your use case, look at black-box solutions like IBM Db2 AI for z/OS (Db2zAI), IBM zSystems Operations Analytics (IZOA) or IBM Db2 13 for z/OS SQL Data Insights (SQL DI).

IBM Cloud Pak for Data (CP4D) is another offering that is instrumental to the future of AI solutions on IBM zSystems. CP4D runs on the IBM zSystems infrastructure in addition to other platforms that are based on Red Hat OpenShift, and it is deeply integrated with WMLz. Typical use cases involve performing model development, training, and bias checks on CP4D, and the model deployment and scoring on WMLz.

For more information about CP4D, see [IBM Cloud Pak for Data](#). For more information about the potential use cases that leverage the strengths of the IBM zSystems infrastructure, see [Capabilities on Linux on IBM zSystems and IBM LinuxONE](#).

IBM also focuses on AI in their hardware design with the OpenBLAS library usage for AI with single instruction, multiple data (SIMD) (starting with IBM z14®) and recently with leveraging the Integrated Accelerator for AI within the Telum chip for AI acceleration (starting with IBM z16). The new chip has an innovative centralized design, which allows customers to use the full power of the AI processor for AI-specific workloads, which makes it ideal for financial services workloads, such as fraud detection, loan processing, clearing and settlement of trades, anti-money laundering, and risk analysis.

With these innovations, customers are positioned to enhance rules-based fraud detection or use ML, accelerate credit approval processes, improve customer service and profitability, identify which trades or transactions might fail, and propose solutions to create a more efficient settlement process.¹

¹ <https://newsroom.ibm.com/2021-08-23-IBM-Unveils-On-Chip-Accelerated-Artificial-Intelligence-Processor>



A

Installation and configuration pointers

This appendix provides planning, installation, and configuration pointers for several components that are described throughout this publication.

This appendix includes the following topics:

- ▶ “IBM z/OS Container Extensions 2.5” on page 100
- ▶ “Red Hat OpenShift Container Platform” on page 100
- ▶ “IBM Data Virtualization Manager for z/OS” on page 101
- ▶ “IBM Open Data Analytics for z/OS” on page 101
- ▶ “IBM Watson Machine Learning for z/OS 2.4.0” on page 101
- ▶ “IBM Db2 AI for z/OS 1.5.0” on page 102
- ▶ “IBM Db2 for z/OS 13 with SQL Data Insights” on page 102
- ▶ “AI on IBM zSystems” on page 103
- ▶ “ONNX resources” on page 103
- ▶ “TensorFlow resources” on page 103
- ▶ “IBM CICS Transaction Server for z/OS resources” on page 103
- ▶ “IBM zSystems Operations Analytics” on page 104
- ▶ “IBM Cloud Pak for Data” on page 104

IBM z/OS Container Extensions 2.5

IBM z/OS Container Extensions (zCX) can be evaluated before purchasing the full feature code by using the limited-time no-charge trial. The trial lasts 90 days and requires an IBM z14 or later. For more information about the trial, including how to set it up, the next steps after the trial ends, and encountering trial errors, see [zCX Trial](#).

For more information about technical details and resources regarding hardware and software requirements and considerations for your system when planning to deploy a zCX instance, see [Planning for zCX](#).

Additionally, the IBM zSystems and LinuxONE Content Solutions provide a zCX content solution web page that describes zCX functions, business benefits that can be derived from the solution, and technical resources to help you get started with using zCX in your environment. The following list briefly describes the steps that are outlined in the content solution that are required to get started with zCX:

1. Research and decide whether zCX is the correct solution for your IBM z/OS system.
2. Prepare your system and plan for your zCX instance. These steps are about how to decide on the allocations and variables that are needed for your zCX instances.
3. Provision your zCX instance by using an IBM z/OS Management Facility (z/OSMF) workflow.
4. Deploy a Linux container in your zCX instance. Log in to zCX and deploy a Docker container by using the Docker command-line interface (CLI).
5. Manage and maintain your zCX instance. z/OSMF has additional workflows to manage the lifecycle of your zCX appliance, including a provisioning workflow, a backup configuration workflow, a reconfiguration workflow, a restore configuration workflow, an add data disks workflow, an upgrade workflow, a rollback workflow, and a deprovisioning workflow.

You can access the IBM zSystems and LinuxONE Content Solution for zCX at [z/OS Container Extensions \(zCX\)](#).

Red Hat OpenShift Container Platform

For more information about how to install Red Hat OpenShift Container Platform V4 on IBM zSystems, see the following resources:

- ▶ [Chapter 16. Installing with IBM z/VM on IBM zSystems and LinuxONE](#)
- ▶ [Chapter 17. Installing with Red Hat Enterprise Linux KVM on IBM zSystems and LinuxONE](#)
- ▶ [Red Hat OpenShift reference architectures](#)

For more information about how Red Hat OpenShift Container Platform enables cloud-native integration and acceleration with IBM zSystems to bring your artificial intelligence (AI) vision to life, see [Hybrid cloud with IBM zSystems](#).

IBM Data Virtualization Manager for z/OS

For more information and resources about how to install, configure, and verify Data Virtualization Manager for z/OS 1.1.0 (DVM) in your environment, see [Installing IBM Data Virtualization Manager for z/OS](#).

For more information and detailed descriptions about the significant tasks that help you virtualize your data by using DVM for z/OS to securely connect to and access data in your z/OS environment by, see [Getting started with IBM Data Virtualization Manager for z/OS](#).

IBM Open Data Analytics for z/OS

The IBM Open Data Analytics for z/OS (IzODA) 1.1.0 content solution describes the functions and business benefits that are offered by IzODA, such as simplified data analysis among others. The content solution also contains technical resources to help you get IzODA running in your z/OS environment today. The following list briefly describes the steps that are outlined in the content solution that are required to get started with IzODA:

1. Install and configure IzODA and create a user ID.
2. For Apache Spark, customize the directory structure, and configure the client authentication.
3. Configure the network, ports, firewalls, memory, and CPU options.
4. Configure a workload manager (WLM).
5. Set up security authorizations and authorized program facility (APF)-authorize LOAD library data sets.
6. Customize Apache Spark, Mainframe Data Service (MDS), and Anaconda.

The content solution also identifies specific roles and corresponding tasks for each role in that process to get started, such as the z/OS System Programmer, the System/Network Administrator, and the Security Administrator. You can access the IBM zSystems and LinuxONE Content Solution for IzODA at [IBM Open Data Analytics for z/OS \(IzODA\)](#).

For more information about Open Data Analytics for z/OS, see [IBM Open Data Analytics for z/OS](#).

IBM Watson Machine Learning for z/OS 2.4.0

You can find details regarding the configuration steps that are taken for our example IBM Watson Machine Learning for z/OS (WMLz) deployment for the implemented solutions in Chapter 3, “Real-time, in-transaction scoring use case scenarios” on page 37 in the WMLz [configuration.pdf](#). Appendix B, “Additional material” on page 105 describes how to obtain this PDF.

At a high level, the planning steps to implement WMLz should include the following ones:

1. Review the architecture requirements and install any prerequisite hardware and software, which are available at [Overview of WML for z/OS](#).
2. Decide on an installation option.

For more information about choosing the best installation option for your needs, see [Choosing an installation option](#).

3. After you review the architecture requirements and select an installation option, verify or install any prerequisite hardware and software for your chosen installation option.

For more information about these prerequisites, see [Installing prerequisite hardware and software for WML for z/OS base](#).

4. Begin the roadmap for planning, installing, and configuring WMLz.

This high-level roadmap outlines and describes the steps to manage, coordinate, and track all major activities for the planning, installation, and configuration of WML for z/OS, and includes the following major components:

- Sequence of steps in the process
- Task and instructions for the task
- Priority (optional or non-optional)
- Type of task (planning, installation, or configuration)
- IT roles and skills (suggested IT role and skill set required to perform the task).

The roadmap can be accessed at [Roadmap for installing and configuring WML for z/OS](#).

The following list contains more WMLz resources to help with your integration and usage of the product within your existing infrastructure:

- ▶ Algorithm, data source, data type, and model type support in WMLz

For more information about WMLz supported algorithms, data sources, data types, and model types, see [Supported algorithms, data sources, data types, and model types](#).

- ▶ Open Neural Network Exchange (ONNX) compiler service for WMLz base

For more information about how to configure the ONNX compiler service for your WMLz base to import, deploy, and manage your ONNX models with WMLz for z/OS base, see [Configuring the ONNX compiler service for your WML for z/OS base](#).

- ▶ Online scoring with IBM Customer Information Control System (IBM CICS) in WMLz

For more information about the integration of WMLz base scoring services in a CICS region for the online scoring of models in WMLz by using the ALNSCORE program, see [Preparing a model for online scoring with CICS program ALNSCORE](#).

- ▶ WMLz and IBM Watson OpenScale

For more information about how to integrate your WMLz base with the IBM Watson OpenScale service on IBM Cloud Pak for Data (CP4D) to validate models, mitigate bias, and explain results, see [Validating a model with IBM Watson OpenScale](#).

IBM Db2 AI for z/OS 1.5.0

For more information about and the procedure for installing and configuring IBM Db2 AI for z/OS (Db2ZAI), see the [Installing and configuring the IBM Db2 AI for z/OS solution](#).

IBM Db2 for z/OS 13 with SQL Data Insights

For more information about IBM Db2 for z/OS 13, see [What is Db2 for z/OS?](#)

For more information about leveraging IBM Db2 13 for z/OS SQL Data Insights (SQL DI), see [Running AI queries with SQL Data Insights](#).

Both Db2 V13 and the SQL DI functions also are described in *IBM Db2 13 for z/OS and More*, SG24-8527.

AI on IBM zSystems

For more information about use cases, code samples, and other content that serves as a technical resource for your AI on IBM zSystems journey, see [AI on IBM zSystems 101](#).

ONNX resources

For more information about how to configure the ONNX compiler service for your WMLz base, see [Configuring the ONNX compiler service for your WML for z/OS base](#).

For more information about the ONNX community-provided tools to help create, optimize, and deploy deep learning (DL) models, see [ONNX supported tools](#).

For more information about ONNX converters, what they are, how they are used, and what converting libraries are available for converting AI models into ONNX, see [ONNX Converters](#).

TensorFlow resources

For more information about the benefits of using TensorFlow framework on IBM zSystems and LinuxONE, see [TensorFlow on IBM zSystems and LinuxONE](#).

You can access the TensorFlow, TensorFlow Serving, and IBM zSystems Optimized for TensorFlow container images for IBM zSystems and LinuxONE at the [IBM zSystems and LinuxONE Container Registry](#).

For more information about how to download and use the IBM zSystems Optimized for TensorFlow container image to transparently target the IBM zSystems Integrated Accelerator for AI for several compute-intensive operations during inferencing with no changes to the TensorFlow models, see the [IBM zSystems Optimized for TensorFlow Documentation and Samples GitHub repository](#).

For educational resources that focus on the TensorFlow framework, including code samples, tutorials, and implementations, see [IBM Developer: About TensorFlow](#).

For more information about installing TensorFlow 2 on your specific system, see [Install TensorFlow 2](#).

IBM CICS Transaction Server for z/OS resources

For more information about CICS, including setting it up in your own environment, select the appropriate version distribution for your system at [IBM CICS Transaction Server for z/OS product documentation](#).

IBM zSystems Operations Analytics

For more information about IBM zSystems Operations Analytics, its capabilities, components, and other key concepts, see [IBM zSystems Operations Analytics overview](#).

IBM Cloud Pak for Data

For more information about the CP4D product offering, see [IBM Cloud Pak for Data](#).

For more information about CP4D on the IBM zSystems infrastructure and potential use cases that leverage the platform strengths, see [Capabilities on Linux on IBM zSystems and IBM LinuxONE](#).



Additional material

This paper refers to additional material that can be downloaded from the internet, as described in the following sections.

Locating the web material

The web material that is associated with this paper is available in softcopy on the internet from the IBM Redbooks web server:

<ftp://www.redbooks.ibm.com/redbooks/REDP5661>

Alternatively, you can go to the IBM Redbooks website:

ibm.com/redbooks

Search for REDP-5661, select the title, and then click **Additional materials** to open the directory that corresponds with the IBM Redpaper form number, REDP5661.

Using the web material

The additional web material that accompanies this paper includes the following files. The AddM.zip compressed file contains all the other files.

| <i>File name</i> | <i>Description</i> |
|----------------------------|--|
| CICS Application-CRED.docx | Document that includes steps to create CICS resources and run the application. Also includes source codes. |
| CREDFILE.JCL | JCL for creating a file that includes sample credit (application data - small). |
| CREDFILE1.JCL | JCL for creating a file that includes sample credit (application data - large). |

| | |
|------------------------------------|--|
| CreditRiskNum Notbeook.ipynb | Data science Notebook that is used to create a credit risk model. |
| CreditRiskNum Notbeook.pdf | PDF of the Notebook that used to create a credit risk model. |
| CREDRSCR.CBL | Credit risk score prediction COBOL program. |
| DFHOWBC2.CBL | Program to make a Representational State Transfer (REST) application programming interface (API) call to a credit risk model. |
| ccf_lstm_to_onnx_RB.ipynb | Data science Notebook showing how to reload a TensorFlow model from saved checkpoint weights, convert the model to an Open Neural Network Exchange (ONNX) format, change the input/output shape of the model to only one dynamic dimension to ensure that micro-batching in an IBM Watson Machine Learning for z/OS (WMLz) deployment can be used, which creates some test inputs to validate that micro-batching works as expected. |
| ccf_220_keras_lstm_static-0S.ipynb | Data science Notebook showing how the fraud detection model is trained and converted to ONNX format. |
| fraud-detection-sample-json.txt | Sample JSON data that is used when invoking the fraud detection model by using the WMLz User Interface (UI) or the IBM Customer Information Control System (IBM CICS) COBOL application. |
| WMLz configuration.pdf | Our WMLz 2.4 installation and configuration details. |

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material AddM.zip file into this folder.

Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this paper.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topics in this document. Some publications that are referenced in this list might be available in softcopy only.

- ▶ *Accelerating Digital Transformation on Z Using Data Virtualization*, REDP-5523
- ▶ *Art of the Possible with AI on IBM zSystems: A Collection of Enterprise Experiences*, CRSE0402
- ▶ *Become Data Driven with IBM Z Infused Data Fabric*, REDP-5680
- ▶ *Demystifying Data with AI on IBM Z*, REDP-5633
- ▶ *Enabling Real-time Analytics on IBM z Systems Platform*, SG24-8272
- ▶ *Getting started with z/OS Container Extensions and Docker*, SG24-8457
- ▶ *IBM Data Virtualization Manager for z/OS*, SG24-8514
- ▶ *IBM Db2 13 for z/OS and More*, SG24-8527
- ▶ *IBM z/OS Container Extensions (zCX) Use Cases*, SG24-8471
- ▶ *IBM z16 (3931) Technical Guide*, SG24-8951
- ▶ *IBM z16 Configuration Setup*, SG24-8960
- ▶ *IBM z16 Technical Introduction*, SG24-8950
- ▶ *IBM Z Connectivity Handbook*, SG24-5444
- ▶ *Installing and Configuring IBM Db2 AI for IBM z/OS v1.4.0*, REDP-5643
- ▶ *Real-time Fraud Detection Analytics on IBM System z*, SG24-8066
- ▶ *Turning Data into Insight with IBM Machine Learning for z/OS*, SG24-8421
- ▶ *What AI Can Do for You: Use Cases for AI on IBM Z*, REDP-5679

You can search for, view, download, or order these documents and other Redbooks, Redpapers, web docs, drafts, and additional materials at the following website:

ibm.com/redbooks

Online resources

The following websites also are relevant as further information sources:

- ▶ AI Model Lifecycle Management Overview blog:
<https://www.ibm.com/cloud/blog/ai-model-lifecycle-management-overview>
- ▶ Announcing Anaconda for Linux on IBM zSystems and LinuxONE:
<https://www.ibm.com/blogs/systems/announcing-anaconda-for-linux-on-ibm-z-linuxone/>
- ▶ Artificial intelligence (AI) on IBM zSystems: Examples of model conversion to Open Neural Network Exchange (ONNX) format:
<https://github.com/IBM/ai-on-z-samples/tree/main/>
- ▶ *COMPUTING MACHINERY AND INTELLIGENCE*, found at:
<https://redirect.cs.umbc.edu/courses/471/papers/turing.pdf>
- ▶ The Data and AI on IBM zSystems portfolio at the nexus of your hybrid cloud:
<https://www.ibm.com/resources/guides/data-ai-hybridcloud-on-ibmz/>
- ▶ Data fabric overview and details:
<https://www.ibm.com/analytics/data-fabric>
- ▶ IBM CICS overview:
<https://www.ibm.com/it-infrastructure/z/cics>
- ▶ IBM Cloud Pak for Data 4.5.x - Virtualizing data with Data Virtualization:
<https://www.ibm.com/docs/en/cloud-paks/cp-data/4.5.x?topic=data-virtualizing>
- ▶ IBM Cloud Pak for Data Deployments:
<https://www.ibm.com/products/cloud-pak-for-data/deployment-model-options>
- ▶ IBM Cloud Pak for Data on IBM zSystems:
<https://www.ibm.com/downloads/cas/YBL2KLER>
- ▶ IBM Data Virtualization Manager for z/OS product documentation:
<https://www.ibm.com/products/data-virtualization-manager-for-zos>
- ▶ IBM Db2 AI for z/OS product page:
<https://www.ibm.com/products/db2-ai-for-zos>
- ▶ IBM Developer on TensorFlow:
<https://developer.ibm.com/components/tensorflow/>
- ▶ IBM Open Data Analytics for z/OS product documentation:
<https://www.ibm.com/products/open-data-analytics-for-zos>
- ▶ IBM Telum processor blog:
<https://www.ibm.com/blogs/systems/ibm-telum-processor-the-next-gen-microprocessor-for-ibm-z-and-ibm-linuxone/>
- ▶ IBM Watson Machine Learning for z/OS overview:
<https://www.ibm.com/products/machine-learning-for-zos>
- ▶ IBM Watson Machine Learning for z/OS product documentation:
<https://www.ibm.com/products/machine-learning-for-zos>

- ▶ IBM zSystems Content Solutions: z/OS Container Extensions
<https://www.ibm.com/support/z-content-solutions/container-extensions/>
- ▶ IBM zSystems and LinuxONE Container Registry:
<https://ibm.biz/zregeap>
- ▶ IBM zSystems in a hybrid cloud world:
<https://www.linkedin.com/pulse/ibm-z-hybrid-cloud-world-c%C3%BCneyt-g%C3%B6ksu/>
- ▶ IBM z/OS Operating Systems Overview:
<https://www.ibm.com/it-infrastructure/z/zos>
- ▶ IBM zSystems Operations Analytics product documentation:
<https://www.ibm.com/products/z-anomaly-analytics>
- ▶ Installing a Red Hat OpenShift cluster with z/VM on IBM zSystems and LinuxONE:
https://docs.openshift.com/container-platform/4.9/installing/installing_ibm_z/installing-ibm-z.html
- ▶ Linux on IBM zSystems overview:
<https://www.ibm.com/it-infrastructure/z/os/linux>
- ▶ ONNX-Multi-Level Intermediate Representation (MLIR) open source project GitHub:
<https://github.com/onnx/onnx-mlir>
- ▶ ONNX website:
<https://onnx.ai/>
- ▶ Red Hat OpenShift Container Platform on IBM zSystems and LinuxONE Reference Architecture:
<https://www.ibm.com/docs/en/linux-on-systems?topic=architecture-platform>
- ▶ Software for enterprise analytics and machine learning:
<https://www.ibm.com/analytics/data-and-ai-on-ibm-z>
- ▶ What's new in WML for z/OS 2.3.0:
<https://www.ibm.com/docs/en/wml-for-zos/2.3.0?topic=zos-whats-new>
- ▶ *What is Artificial Intelligence?*, found at:
<http://jmc.stanford.edu/articles/whatisai/whatisai.pdf>
- ▶ What is Db2 for z/OS:
<https://www.ibm.com/docs/en/db2-for-zos/12?topic=getting-started-db2-zos>
- ▶ What is z/OS Container Extensions:
<https://www.ibm.com/docs/en/zos/2.4.0?topic=extensions-what-is-zos-container>

Help from IBM

IBM Support and downloads

[ibm.com/support](https://www.ibm.com/support)

IBM Global Services

[ibm.com/services](https://www.ibm.com/services)

Abbreviations and acronyms

| | | | |
|---------------|--|---------------|---------------------------------------|
| AI | artificial intelligence | NLP | natural language processing |
| AML | anti-money laundering | NLU | natural language understanding |
| APF | authorized program facility | OLAP | online analytical processing |
| API | application programming interface | OLTP | online transaction processing |
| AutoML | automated machine learning | ONNX | Open Neural Network Exchange |
| CADS | Cognitive Assistant for Data Scientists | opset | operation set |
| CI/CD | continuous integration and continuous delivery (CI/CD) | OSCE | Online Scoring Community Edition |
| CICS | IBM Customer Information Control System | PMML | Predictive Model Markup Language |
| CLI | command-line interface | PoC | proof of concept |
| CNN | convolutional neural network | PTF | product temporary fix |
| CODAIT | Center for Open-source Data and AI Technologies | ReLU | Rectified Linear Unit |
| CP4D | Cloud Pak for Data | REST | Representational State Transfer |
| Db2ZAI | IBM Db2 AI for z/OS | RNN | recurrent neural network |
| DL | deep learning | SCP | Secure Copy Protocol |
| DNN | deep neural network | SIMD | single instruction, multiple data |
| DR | disaster recovery | SLA | service-level agreement |
| DVM | IBM Data Virtualization Manager for z/OS | SMF | System Management Facility |
| FL | function level | SQL | Structured Query Language |
| HA | high availability or highly available | SQL DI | IBM Db2 13 for z/OS SQL Data Insights |
| HPO | hyper parameter optimization | SU | service unit |
| IBM | International Business Machines Corporation | UI | user interface |
| IDAA | IBM Db2 Analytics Accelerator | WLM | workload manager |
| IDE | integrated development environment | WMLz | IBM Watson Machine Learning for z/OS |
| ITOA | IT Operational Analytics | YOLO | You Only Look Once |
| IZOA | IBM zSystems Operations Analytics | z/OSMF | z/OS Management Facility |
| IzODA | IBM Open Data Analytics for z/OS | zCX | z/OS Container Extensions |
| JDBC | Java Database Connectivity | zDLC | IBM zSystems Deep Learning Compiler |
| JVM | Java virtual machine | zDNN | IBM zSystems Deep Neural Network |
| KPI | key performance indicator | | |
| LPAR | logical partition | | |
| LSTM | long short-term memory | | |
| MDS | Mainframe Data Service | | |
| ML | machine learning | | |
| MLIR | Multi-Level Intermediate Representation | | |



REDP-5661-01

ISBN 0738460923

Printed in U.S.A.

Get connected

