A white background with text and images

AI-generated content may be incorrect.

**Table of Contents**

[1.0 Overview and Goals of Lab 1](#_Toc206274913)

[1.1 Overview 1](#_Toc206274914)

[1.2 Goals 1](#_Toc206274915)

[2.0 Lab prerequisites 2](#_Toc206274916)

[3.0 Machine Learning for z/OS 7](#_Toc206274917)

[3.1 Login to Machine Learning for z/OS (MLz) 7](#_Toc206274918)

[3.2 The IBM MLz Main Dashboard 8](#_Toc206274919)

[4.0 Import a Model into MLz 9](#_Toc206274920)

[4.1 MLz sample model 9](#_Toc206274921)

[4.2 Click on the Models tab on the MLz main dashboard 9](#_Toc206274922)

[4.3 Import model into MLz 10](#_Toc206274923)

[5.0 Deploy Your Model 13](#_Toc206274924)

[5.1 Using the Action Pulldown Menu 13](#_Toc206274925)

[5.2 Selecting the Deploy option 14](#_Toc206274926)

[5.3 Completing the Create deployment form 15](#_Toc206274927)

[5.4 Finding the model deployment ID 16](#_Toc206274928)

[6.0 Use Python to Access Scoring Endpoint via Jupyter Notebook 18](#_Toc206274929)

[6.1 Sign In to Jupyter Notebook 19](#_Toc206274930)

[6.2 Upload the sample python program to Jupyter Notebook 19](#_Toc206274931)

[6.3 Using Jupyter Notebook to load cars.ipynb file that contains our initial python code 20](#_Toc206274932)

[6.4 Executing Cells in Jupyter Notebook 21](#_Toc206274933)

[6.5 Updating code to talk to our scoring endpoint 22](#_Toc206274934)

[7.0 Summary 23](#_Toc206274935)

# 1.0 Overview and Goals of Lab

This lab will explore the process of importing and deploying a model onto z/OS using that model in real-time from a python program via the scoring server endpoint.

## 1.1 Overview

IBM Machine Learning for z/OS (MLz), the flagship AI platform for z/OS:

* Provides a GUI that allows for model management, deployment, and administration
* Supports hosting models using many different runtimes including Spark ML, Python ML, PMML scoring, and ONNX scoring
* Provides a model repository database, **RESTful endpoints for inference calls**, high-performance integrated scoring capabilities for CICS and IMS applications, and a Jupyter Notebook environment
* Provides micro-batching capabilities to maximize inferencing efficiency by allowing for multiple requests to be bunched together
* Will also optimize the model to leverage the new Integrated Accelerator for Analytics (AIU) whenever possible and if available

MLz allows z/OS applications to interact with models all running directly within the z/OS ecosystem.

We will be using the MLz 3.2 Enterprise in our lab today.

## 1.2 Goals

1. Import the **cars-test-model.tar.gz** model into MLz
2. Deploy the **cars-test-model.tar.gz** model into MLz and obtain the scoring server endpoint.
3. Use a python script via Jupyter Notebook running on z/OS to query the scoring server for the model and quickly obtain the result from the model
   1. Use the REST API interface and JSON to interact with the scoring server endpoint

# 2.0 Lab Prerequisites

We will be using a **remote desktop** environment to access our z/OS system located behind an IBM firewall.

To access the **remote desktop**, you will need to enter the following URL:

<https://techzone.ibm.com/my/workshops/student/689f829646c8a302aef248d9>

**A screenshot of a computer

AI-generated content may be incorrect.**

In the text field for the password/access code, enter: **aionz**

Now, you will be in a screen that displays your assigned environment:

A screenshot of a computer

AI-generated content may be incorrect.

Scroll down until you locate the blue rectangle under the text VM Remote Console,

A screenshot of a computer

AI-generated content may be incorrect.

To start your remote desktop session, click on the link in the blue rectangle.

You will see a black screen pop-up and on top of the black area you will see buttons for Reboot, Shutdown, and Ctrl+Alt+Del in red:

A screenshot of a phone

AI-generated content may be incorrect.

Click on the Ctrl+Alt+Del red button:

A screenshot of a computer screen

AI-generated content may be incorrect.

Now, you will see the Windows remote desktop. By default, the Administrator username is selected, and you will be prompted for a password.

Enter the password, **IBMDem0s**

A screenshot of a computer

AI-generated content may be incorrect.

In your remote desktop, in the Windows Search text area, type Cisco

A computer screen with a blue background

AI-generated content may be incorrect.

Now, click on Cisco Secure Client, the Cisco Secure Client application will start,

A computer screen with a lock and text

AI-generated content may be incorrect.

If Cisco AnyConnect was previously set up for you, you will already see the correct server, **asa003b.centers.ihost.com**, in the field next to the **Connect** button. In that case, click on Connect to establish the VPN connection to IBM.

\*\*\* **If successfully connected, you can skip the Cisco AnyConnect set up and move on to the “3.0 Machine Learning for z/OS” section. \*\*\***

If the server next to the Connect button is not **asa003b.centers.ihost.com**, replace it with: a**sa003b.centers.ihost.com**

Click Connect after replacing the VPN site name with **asa003b.centers.ihost.com** to view the Cisco Any Connect sign-on window that requests the username and password,

A screenshot of a computer

AI-generated content may be incorrect.

Use the username and password provided for access to the IBM VPN network. **It is different from the MLz username and password.**

After successfully establishing a VPN connection, you can proceed to the next section.

**NOTE:** **The files needed for this workshop including this handout, the cars-test-model.tar.gz, and the cars.ipynb are available in the provided gitHub repository.**

You need to access the gitHub repository from the remote desktop environment.

In a browser window in your remote desktop environment, enter the following URL to download the material for this lab:

<https://www.github.com/WalterManrique/cars-weight-model-onz>

A screenshot of a computer

AI-generated content may be incorrect.

Click on the cars-test-model.tar.gz link to download the model.

After clicking on the link for the cars-test-model.tar.gz, click on the button with the ellipse to display the window with the Download option and click on Download.

A screenshot of a computer

AI-generated content may be incorrect.

To download the cars.ipynb Jupyter Notebook file clink on the respective link and follow the previous gitHub steps.

# 3.0 Machine Learning for z/OS

## 3.1 Log in to Machine Learning for z/OS (MLz)

Let’s log in to MLz from our remote desktop environment and get ready to **import** our model. In your remote desktop environment, open a web browser. In the browser’s URL field, type the following:

<https://129.40.117.161:9888>

After entering the URL in your browser, you will end up in the following MLz Log in screen:

A screenshot of a computer

AI-generated content may be incorrect.

NOTE:

The username for this lab has the following format: **aizos0xx** where **xx** are digits and the password is **z17aionz** all in lowercase. You should have received a paper with your MLz login credentials.

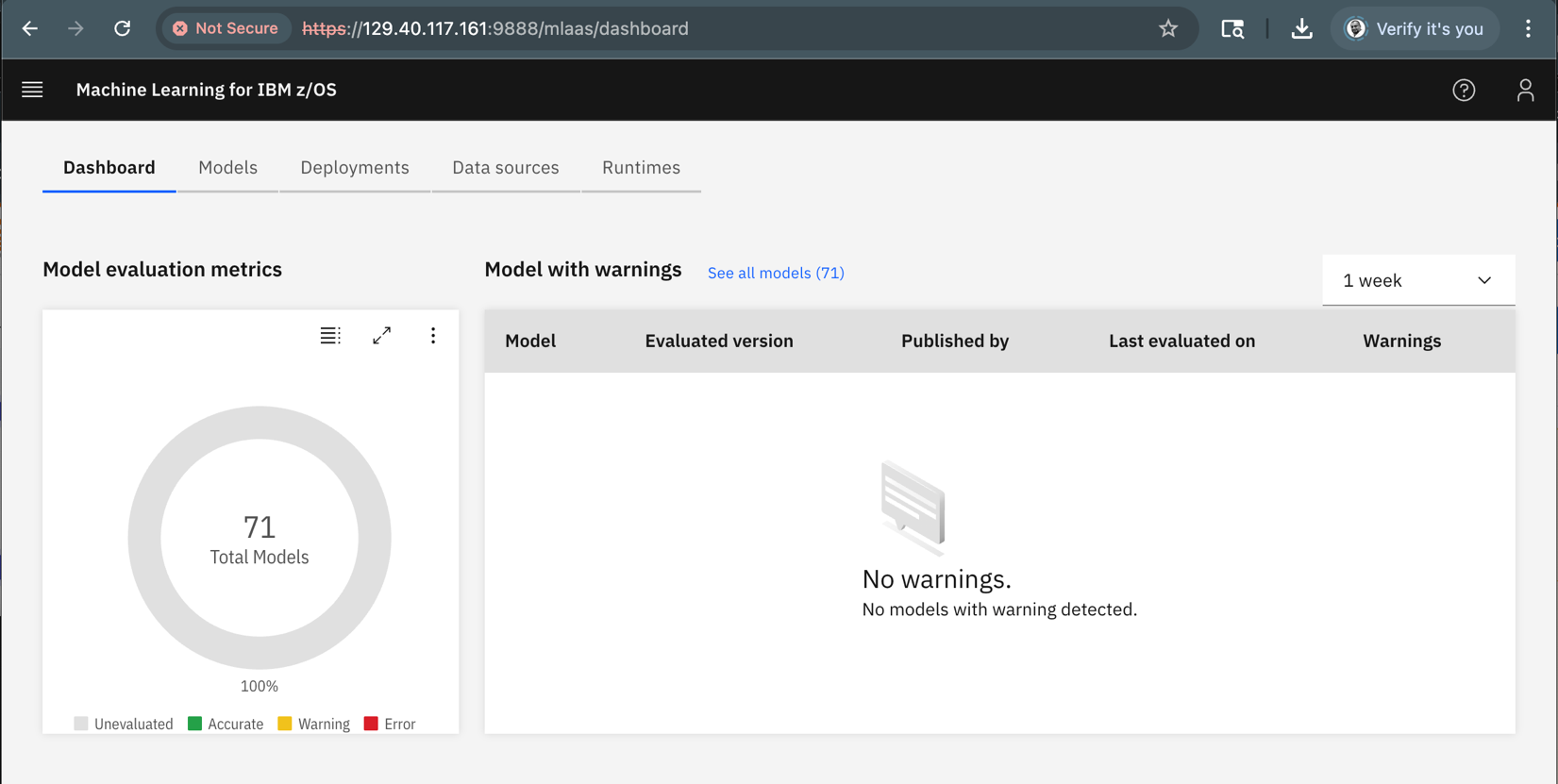
Enter your MLz login credentials for this lab all in lowercase in the respective fields and click the **Log in** button.

## 3.2 The IBM MLz Main Dashboard

## 

After you log in, the main dashboard for MLz will be displayed. The MLz dashboard is a web-based interface used for managing and administering the MLz platform.

The main dashboard provides the tools necessary for model management, user management, and system administration.



# 4.0 Import a Model into MLz

## 4.1 MLz Sample Model

The model that we will use in this workshop is a sample model that is included in MLz 3.2 and it can be found in the $IML\_INSTALL\_DIR/alnsamp directory under z/OS. The **cars-test-model.tar.gz** file contains the sample model. It is a sample **regression** model that can be used to predict the weight of a car.

By now, you should have already downloaded the model to your remote desktop environment. It should be in the same folder together with the **cars.ipynb** program that we will use later in the workshop.

## 4.2 Displaying the Models

Click on the Models tab on the MLz main dashboard.

You will then be presented with the Models Table Screen.

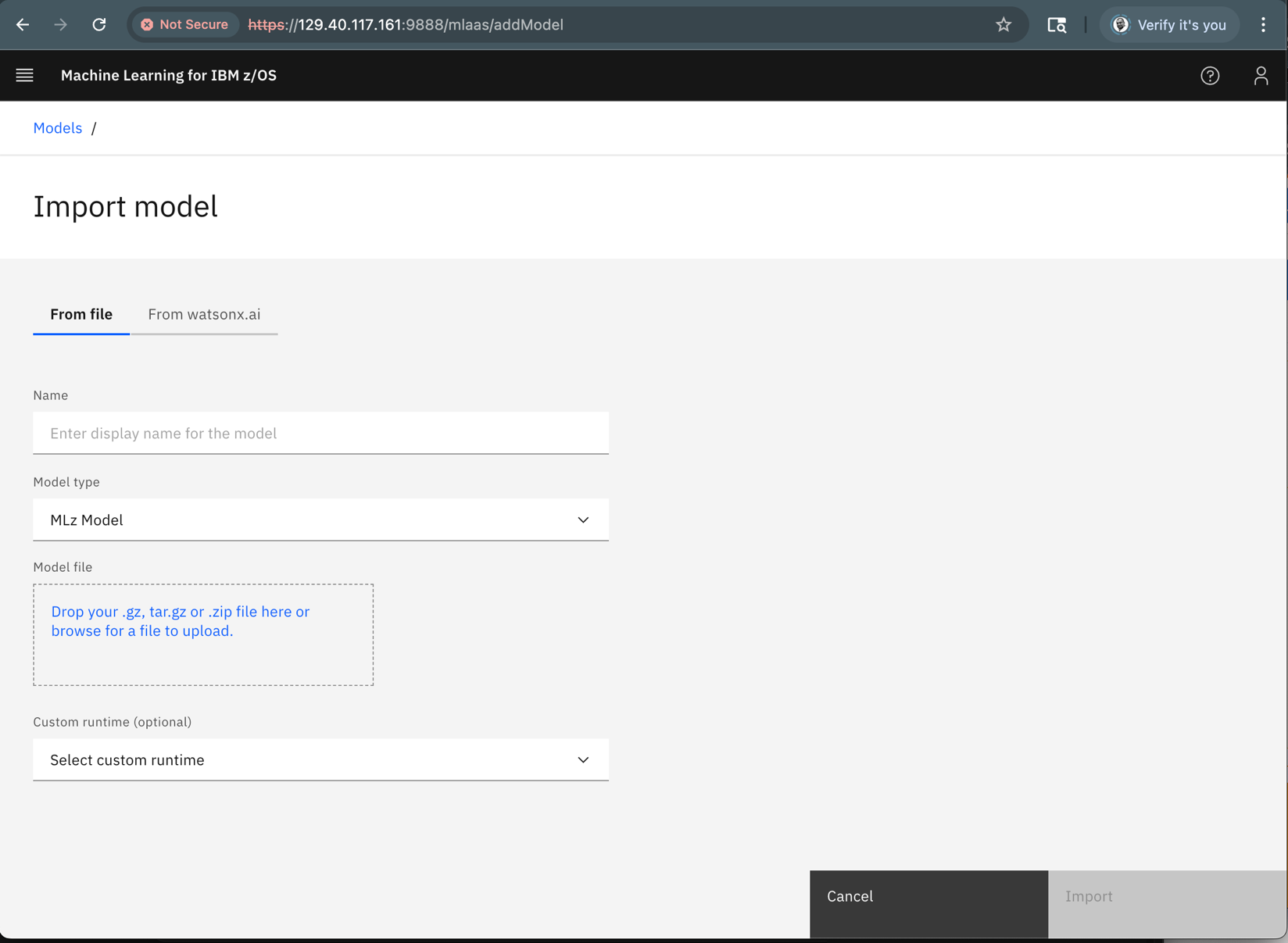
A screenshot of a computer

AI-generated content may be incorrect.

The Models Table Screen has blue button with the “Import model” text. We will click on it to import our model in the next section.

## 4.3 Import model into MLz

Click on the blue “Import model” button to navigate to the Import Model screen.



* On this screen, click the “**From file”** tab if not already selected
* Enter the name for the new model in the **Name** field. To avoid name clashing with other lab participants, we suggest naming your model **cars<nn>,** where <nn> represents the last two digits of the **aizos0*xx*** userid that you used to login to MLz earlier. For example, if your userid was aizos001, you would name your model cars01
* There is no need to update the **Model type** field
* Drag the file named **cars-test-model.tar.gz** from the Downloads folder to the ‘Model file” field to upload the model source file to MLz. Alternatively, you can click the blue text in the **Model file** box and use the interactive file explorer to search the file manually.

After completing all the previous steps, your screen should look something like this:

A screenshot of a computer

AI-generated content may be incorrect.

Note that the Import button at the bottom right of the browser is now blue.

Click the **Import** blue button in the bottom right-hand corner to import the selected model.

Importing takes the model, compiles it and generates z/OS executable code using the best hardware available.

Once the model is imported, the browser will display the list of models. The latest imported model will be listed on top:

A screenshot of a computer

AI-generated content may be incorrect.

Verify that your model is listed and no errors are reported. There will be a green check mark under the status column if there were no errors.

Now, you are ready to deploy your newly imported model.

# 5.0 Deploy Your Model

By now, you have imported a model into MLz. The model shows a green check mark under the status column which means it is ready to be deployed.

The model needs to be deployed to a scoring service managed by MLz. You can think of the scoring service as the location where the model will run.

There are different types of scoring services and depending on the use case for the deployment of your models, you can have different scoring service configurations. The scoring service can be configured:

* on a single z/OS image running in a Liberty server managed by MLz
* for High Availability (HA), which gives it the ability to leverage DDVIPAs on the system to round-robin the inferencing requests across multiple LPARs in your SYSPLEX
* to run inside a Liberty server within a CICS region, configured for normal or HA
* and much more…

Let’s deploy our model. From the list of models, find your model, and find the **Action** column on the right.

## 5.1 Using the Action Pulldown Menu

Click on the 3 stacked dots (vertical ellipsis) to get a list of actions that we can perform on this model.

A screenshot of a computer

AI-generated content may be incorrect.

## 5.2 Selecting the Deploy Option

Select **Deploy** from the pulldown menu.

You will now be taken to the **Create Deployment** page.

A screenshot of a computer

AI-generated content may be incorrect.

## 5.3 Completing the Create Deployment Form

Fill in the following values:

* For **Deployment name**, use the same naming convention as before with your two number suffix, concatenating that number at the end of the word cars.

E.g., **cars<nn>.** If you were assigned, aizos001, this value would be **cars01.**

* For **Deployment type**, the value will be **Online**.
* For **Model version**, there should be only one, so the only selectable value will be **1.**
* **For Scoring service (standalone or cluster),** we are going to deploy this model into a Liberty server that leverages the **WebSphere Optimized Local Adapter** (WOLA) high-speed communications. Select **WOLASCOR** as our scoring service. If not available, select the only available option.
* There are some other deployment options but those are beyond the scope of this lab.
* Once you have filled in all the fields, click **Create** in the bottom right-hand corner to deploy the model.

A screenshot of a computer

AI-generated content may be incorrect.

Let’s verify that the model is deployed and that cars<nn> is listed under the Deployments tab.

A screenshot of a computer

AI-generated content may be incorrect.

## 5.4 Finding the Model Deployment ID

Each model deployed is given a unique deployment ID that is used by the model invoker application to tell MLz which deployed model to use for the scoring request. We will need the deployment model ID when we invoke the model from our python program in the following steps.

To find the deployment ID, click on the name of your deployed model on the deployments page.

A screenshot of a computer

AI-generated content may be incorrect.

After clicking on the deployed model, you will end up in the deployment details page which includes a scoring endpoint. The last qualifier of the scoring endpoint URL is deployment ID.

A screenshot of a computer

AI-generated content may be incorrect.

Copy the **complete** endpoint value by **clicking** on the **copy icon** next to the scoring point URL and save it.

A screenshot of a computer

AI-generated content may be incorrect.

You can save the complete scoring point URL in a notepad document, or you can come back to the deployment page to copy the scoring point URL when needed.

# 6.0 Use Python to Access Scoring Endpoint via Jupyter Notebook

Now, we are ready to communicate with our deployed scoring model via the scoring endpoint using a REST API call. We will write a simple application in python to query the model. Our python program will execute from a Jupyter Notebook running on z/OS.

In your remote desktop environment, navigate to the Jupyter Notebook by entering the following URL in the address field in your browser:

<https://129.40.117.161:9900/>

After entering the URL listed, you will be in the following Sign In page for the jupyterhub.

A screenshot of a computer

AI-generated content may be incorrect.

## 6.1 Sign In to Jupyter Notebook

The username for the Jupyter Notebook is the same as the MLz username: **aizos0xx** where **xx** are digits and the password is **z17aionz** all in lowercase. You should have received a paper with your MLz login credentials. Enter your username and password and click the **Sign In** button.

Initially, after you sign in, your browser will not list any notebooks.

A screenshot of a computer

AI-generated content may be incorrect.

## 6.2 Upload the Sample Python Program to Jupyter Notebook

Click on the upward pointing arrow to upload your cars.ipynb notebook into the z/OS environment. Use the file explorer to select the cars.ipynb file and click on open.

A screenshot of a computer

AI-generated content may be incorrect.

## 6.3 Using Jupyter Notebook to Load cars.ipynb File that Contains our Initial Python Code

Now that the cars.ipynb file has been uploaded to z/OS, a new entry will be displayed under the Name column on the left-hand side of your browser. Double click on the filename to access the code. The code will be displayed on the right-hand side of the browser window.

A screenshot of a computer

AI-generated content may be incorrect.

## 6.4 Executing Cells in Jupyter Notebook

Before we start updating the python code displayed in the Jupyter Notebook, let’s go over one basic concept to avoid a common problem.

The brackets [] are cells within the notebook. The active cell will be executed if the run button is clicked.

In the following example, the active cell is highlighted in blue, if we press run, the active cell will execute bypassing the execution of the previous cell. In other words, none of the import statements would have executed.

A screenshot of a computer

AI-generated content may be incorrect.

Ideally, we will start the execution of our python program from the top cell. Remember to make the top cell the active cell when trying to run your program for the first time. Afterwards, depending on the logic flow, you can repeat the execution of a cell or skip the execution of a cell.

For our lab, let’s always start at the top and execute the cells in the respective order.

## 6.5 Updating Code to Talk to Our Scoring Endpoint

At this point, we are ready!

Let’s follow the instructions on the Jupyter Notebook and update the python program to use our recently deployed model.

Remember to use the complete scoring endpoint for your deployed model found on step 5.4.

A screenshot of a computer

AI-generated content may be incorrect.

# 7.0 Summary

So what did we just do?

1. We imported a model into Machine Learning for z/OS (MLz)
2. We deployed the model into our Liberty server scoring service with WOLA for ultrafast scoring calls
3. We executed a python program from a Jupyter Notebook running on z/OS to query our deployed model using the deployed model’s scoring endpoint via a REST API call
4. We became familiar with Machine Learning for z/OS

Imagine the possibilities you have with a scoring service natively running on z/OS!

We hope you enjoyed our introductory Lab. If you think your company would be interested in a no-charge AI on IBM Z Discovery Workshop, please feel free to drop us a note at aionz@us.ibm.com. The IBM Z Discovery Workshop is a custom workshop experience to learn about how you can start leveraging AI on the zSystems platform.