



AI on IBM Z

# Anti-money laundering solution template

This solution template provides an example on how to deploy AI using an IBM Z environment, while making use of open source frameworks, Machine Learning for IBM z/OS (MLz), and more.

Within this solution template, there are various phases of the AI lifecycle included. Work through each of the following steps to deploy your own anti-money laundering solution on IBM Z.



## Table of contents

AI model training.....	3
AI model deployment.....	8
AI model integration.....	12



## Step 1

# AI model training

We will build an anti-money laundering AI model by training with the provided Rapid AI on IBM Z Development Jupyter notebook. Simply point the Jupyter notebook to your dataset and run it to generate your AI model. This trained AI model can then be deployed with MLz.

All sample code for this section is within

```
aionz-st-anti-money-laundering/zST-model-training-jupyter
```

### Prerequisites

- Must have Python 3.9 & 3.11 installed

### Dataset guidance

Sample anti-money laundering dataset can be found on Kaggle -

<https://www.kaggle.com/datasets/ealtman2019/ibm-transactions-for-anti-money-laundering-aml>

### Required features

- Timestamp
- From Bank
- From Account
- To Bank
- To Account
- Amount Received
- Receiving Currency
- Amount Paid
- Payment Currency
- Payment Format
- Is Laundering

## 1. AI model training

Configure rapid AI on IBM Z development environment

Provide data

Data pre-processing

Model training

Access trained AI model

## 2. AI model deployment

# Configure rapid AI on IBM Z development environment

1. Create and activate Python virtual environment

```
python3.9 -m venv env  
source env/bin/activate
```

2. Install required Python packages

```
pip install -r requirements.txt
```

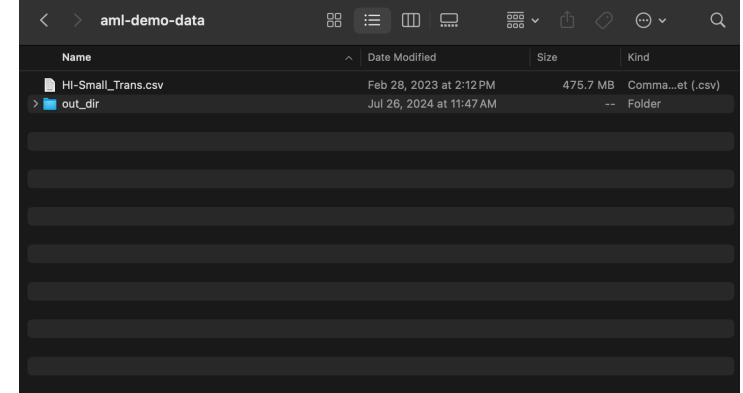
## 4. AI model integration

```
evanlevens@ibp [zst-model-training-jupyter] python3.11 -m venv env  
evanlevens@ibp [zst-model-training-jupyter] source env/bin/activate  
(env) evanlevens@ibp [zst-model-training-jupyter] pip install -r requirements.txt  
Requirement already satisfied: snapml==1.4.2 in /env/lib/python3.11/site-packages (from -r requirements.txt (line 1)) (1.14.2)  
Requirement already satisfied: lightgbm==4.4.0 in /env/lib/python3.11/site-packages (from -r requirements.txt (line 2)) (4.4.0)  
Requirement already satisfied: numpy==1.25.3 in /env/lib/python3.11/site-packages (from -r requirements.txt (line 3)) (1.25.3)  
Requirement already satisfied: pandas in /env/lib/python3.11/site-packages (from -r requirements.txt (line 4)) (2.2.2)  
Requirement already satisfied: scikit-learn==1.1.2 in /env/lib/python3.11/site-packages (from -r requirements.txt (line 5)) (1.1.2)  
Requirement already satisfied: joblib==1.4.1 in /env/lib/python3.11/site-packages (from -r requirements.txt (line 6)) (1.4.1)  
Requirement already satisfied: shap in /env/lib/python3.11/site-packages (from -r requirements.txt (line 7)) (0.46.0)  
Requirement already satisfied: sklearn-gamma in /env/lib/python3.11/site-packages (from -r requirements.txt (line 8)) (0.45.0)  
Requirement already satisfied: python-dateutil==2.8.2 in /env/lib/python3.11/site-packages (from pandas->r requirements.txt (line 9)) (2.9.0.post0)  
Requirement already satisfied: pytz==2028.1 in /env/lib/python3.11/site-packages (from pandas->r requirements.txt (line 10)) (2028.1)  
Requirement already satisfied: numpy-base==1.25.3 in /env/lib/python3.11/site-packages (from -r requirements.txt (line 11)) (1.14.2)  
Requirement already satisfied: joblib==1.4.2 in /env/lib/python3.11/site-packages (from scikit-learn->r requirements.txt (line 5)) (1.4.2)  
Requirement already satisfied: numpy==1.25.3 in /env/lib/python3.11/site-packages (from scikit-learn->r requirements.txt (line 6)) (1.25.3)  
Requirement already satisfied: contourpy==1.4.1 in /env/lib/python3.11/site-packages (from matplotlib->r requirements.txt (line 12)) (1.4.1)  
Requirement already satisfied: cycler==0.1.1 in /env/lib/python3.11/site-packages (from matplotlib->r requirements.txt (line 13)) (0.1.1)  
Requirement already satisfied: kiwisolver==1.4.3 in /env/lib/python3.11/site-packages (from matplotlib->r requirements.txt (line 14)) (1.4.3)  
Requirement already satisfied: packaging==20.8 in /env/lib/python3.11/site-packages (from matplotlib->r requirements.txt (line 6)) (24.1)  
Requirement already satisfied: pyparsing==3.1.2 in /env/lib/python3.11/site-packages (from matplotlib->r requirements.txt (line 15)) (3.1.2)  
Requirement already satisfied: pytz==2028.1 in /env/lib/python3.11/site-packages (from shap->r requirements.txt (line 7)) (2028.1)  
Requirement already satisfied: numpy==1.25.3 in /env/lib/python3.11/site-packages (from shap->r requirements.txt (line 8)) (0.45.0)  
Requirement already satisfied: numba in /env/lib/python3.11/site-packages (from shap->r requirements.txt (line 9)) (0.46.0)  
Requirement already satisfied: cloudpickle in /env/lib/python3.11/site-packages (from shap->r requirements.txt (line 10)) (0.3.0)  
Requirement already satisfied: six==1.16.0 in /env/lib/python3.11/site-packages (from python-dateutil==2.8.2->pandas->r requirements.txt (line 4)) (1.16.0)  
Requirement already satisfied: llvmlite<0.44,>>0.43.0dev0 in /env/lib/python3.11/site-packages (from pandas->r requirements.txt (line 7)) (0.43.0)  
[notice] A new release of pip is available: 23.2.1 >> 24.2  
[notice] To update, run: pip install --upgrade pip
```

## Provide data

1. Put your input dataset (HI-Small\_Trans.csv) in `aml-demo-data/` directory

Note: It is important to put the dataset here because it's referenced by the data pre-processing and model training code (format\_raw\_transactions.py & aml\_model\_training.ipynb)



## 1. AI model training

Configure rapid AI on IBM Z development environment

Provide data

Data pre-processing

Model training

Access trained AI model

## 2. AI model deployment

### Data pre-processing

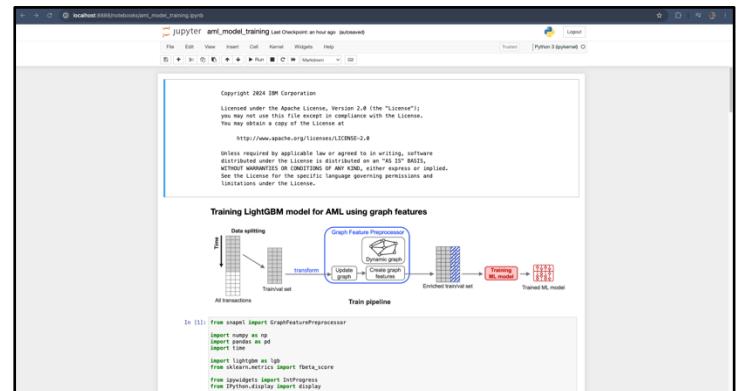
- Run `python3.9 format_raw_transactions.py` to pre-process original dataset into a format suitable for model training (this will take several minutes to complete due to the volume of data, potentially up to 25 mins)

## 4. AI model integration

```
[env] evan@evans-MBP [1st-model-training-jupyter] python3.9 format_raw_transactions.py
Processed 50000 transactions
Processed 100000 transactions
Processed 150000 transactions
Processed 200000 transactions
Processed 250000 transactions
Processed 300000 transactions
Processed 350000 transactions
Processed 400000 transactions
Processed 450000 transactions
Processed 500000 transactions
Processed 550000 transactions
Processed 600000 transactions
Processed 650000 transactions
Processed 700000 transactions
Processed 750000 transactions
Processed 800000 transactions
Processed 850000 transactions
Processed 900000 transactions
Processed 950000 transactions
Processed 1000000 transactions
Processed 1050000 transactions
Processed 1100000 transactions
Processed 1150000 transactions
Processed 1200000 transactions
Processed 1250000 transactions
Processed 1300000 transactions
Processed 1350000 transactions
Processed 1400000 transactions
Processed 1450000 transactions
Processed 1500000 transactions
Processed 1550000 transactions
Processed 1600000 transactions
Processed 1650000 transactions
Processed 1700000 transactions
Processed 1750000 transactions
```

### Model training

- Run Jupyter (using python 3.11.5 kernel)  
`jupyter notebook`
- View Jupyter interface
  - Go to [localhost:8888](http://localhost:8888) in a web browser
- Step through and run all cells within Jupyter notebook (`aml_model_training.ipynb`) within web browser



```
Copyright 2014 IBM Corporation
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Training LightGBM model for AML using graph features

Data splitting
All transactions
Train set
Test set
Data splitting
Graph Feature Preprocessor
Dynamograph
Update graph
Create graph features
Train pipeline
Train ML model
Test ML model

In [1]: from sklearn import GraphFeaturePreprocessor
import numpy as np
import pandas as pd
import time
from lightgbm import LGBMClassifier
from sklearn.metrics import f1_score
from imblearn import ImProgress
from IPython.display import display
```

## 1. AI model training

Configure rapid AI on IBM Z development environment

Provide data

Data pre-processing

Model training

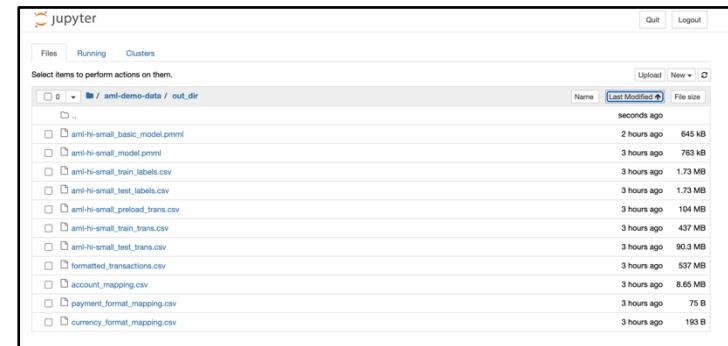
[Access trained AI model](#)

## 2. AI model deployment

### Access trained AI model

- Once training is complete, you can find your AI models within the `aml-demo-data/out_dir/` directory (`aml_hi-small_model.pmml`).

## 4. AI model integration



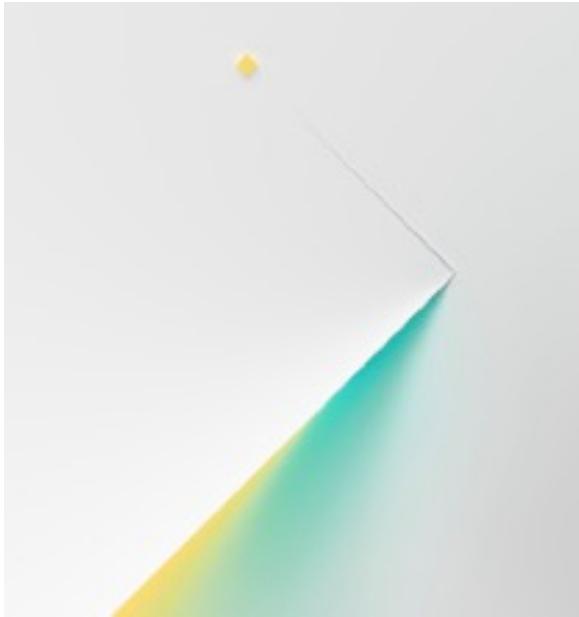
---

1. AI model training

2. AI model deployment

4. AI model integration

AI model training complete



### Prerequisites

- Must have MLz 3.2 installed

### Step 2

## AI model deployment

We will deploy our anti-money laundering AI model using MLz. We can utilize the model import functionality on the MLz UI. This deployed AI model can then be integrated into applications within the IBM Z environment.

1. AI model training

2. AI model deployment

4. AI model integration

[Go to MLz UI](#)

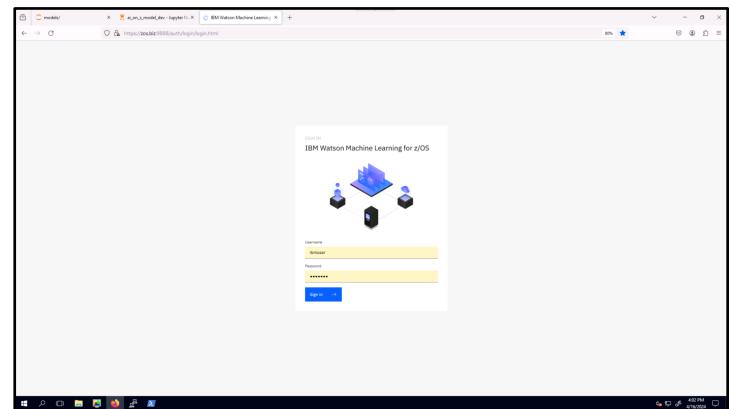
[Import AI model](#)

[Deploy AI model](#)

[View deployed AI model](#)

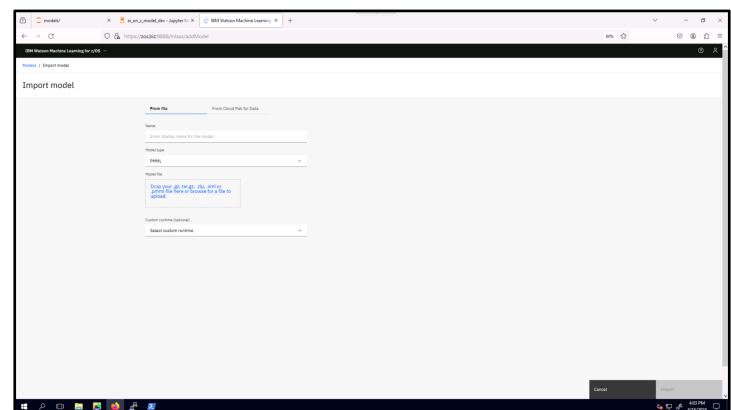
## Go to MLz UI

1. Sign in with username/password



## Import AI model

1. Go to models tab
2. Click import model
3. Enter model name
4. Choose model type
  - a. Choose PMML if using your previously trained model
5. Drag and drop model file  
Use your previously trained model
6. Click import



1. AI model training

2. AI model deployment

4. AI model integration

[Go to MLZ UI](#)

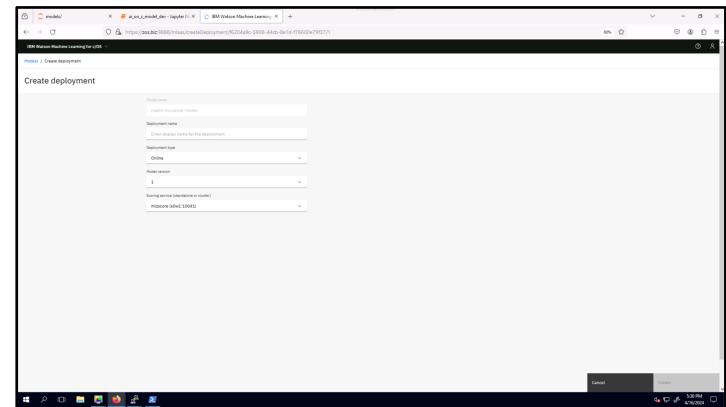
[Import AI model](#)

[Deploy AI model](#)

[View deployed AI model](#)

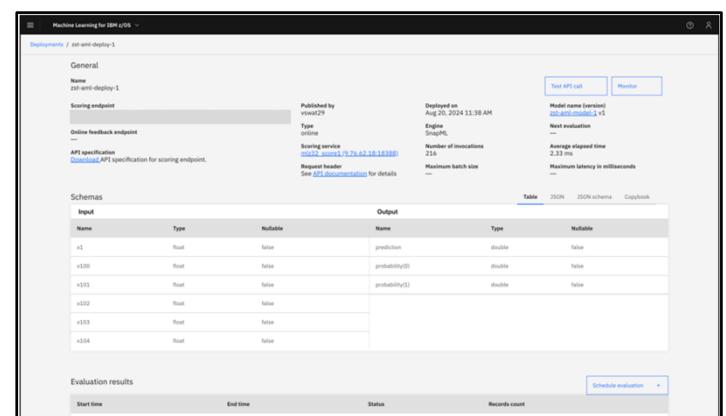
## Deploy AI model

1. Go to models tab
2. Click action button for your model (on right side)
3. Click deploy
4. Enter deployment name
5. Choose deployment type
6. Choose model version
7. Choose scoring service  
Note: you should choose the correct scoring service based on your application (e.g. REST API)
8. Click create



## View deployed AI model

1. Go to deployments tab
2. Click on action button for your deployed model (on right side)
3. Click view details



---

1. AI model training

2. AI model deployment

4. AI model integration

AI model deployment complete



### Prerequisites

- Must have node.js v16 or newer installed
- Must have Docker/podman installed
- Must have Git installed

### Step 3

## AI model integration

We can use our deployed MLz anti-money laundering AI model and integrate it into different types of applications. Guidance on integrating the AI model into a sample anti-money laundering application is below.

All sample code for this section is within

```
aionz-st-anti-money-laundering/zST-model-integration-  
aml
```

## 1. AI model training

[Get model details for inferencing](#)

[Configure sample application](#)

[Build sample application](#)

[Deploy sample application](#)

[Access sample application](#)

## 2. AI model deployment

### Get model details for inferencing

1. Go to MLz UI
2. Go to deployments tab
3. Click on action button for your deployed model (on right side)
4. Click view details
5. Copy scoring endpoint

## 4. AI model integration

The screenshot shows the 'Machine Learning for IBM z/OS' interface with the 'Deployments' tab selected. A deployment named 'zST-anti-deploy-2' is listed. The 'General' section shows the name, published by 'wmlz29', type 'online', and engine 'SnapML'. It also shows the deployment date 'Aug 20, 2024 11:38 AM', number of invocations '216', and maximum batch size '256'. The 'Schemas' section displays input and output schema details. The 'Evaluation results' section shows evaluation metrics like 'Average elapsed time' (2.33 ms) and 'Maximum latency in milliseconds'.

### Configure sample application

1. Set the environment variables within

```
aionz-st-anti-money-laundering/zST-model-integration-aml/env.list
```

USE\_WMLZ (set to true to ensure MLz is used)

WML\_USER (MLz username)

WML\_PASS (MLz password)

WML\_AUTH\_URL (MLz auth token url)

WMLZ\_SCORING\_URL (MLz scoring endpoint)

```
zST_aml > aionz-st-anti-money-laundering > zST-model-integration-aml > env.list
1 USE_WMLZ=true
2 WML_USER=
3 WML_PASS=
4 WML_AUTH_URL=https://<ip>:<port>/auth/generateToken
5 WMLZ_SCORING_URL=https://<ip>:<port>/im/v2/scoring/online/<uuid>
6 TRANSFORMED_DATA_PATH=data/transformed_20k.npy
7 SHAPPATH=data/shapvalues20K.pkl
8 TESTPATH=data/aml-test-transactions.txt
```

## Get model details for inferencing

## Configure sample application

## Build sample application

## Deploy sample application

## Access sample application

## Build sample application

- ## 1. Run command in terminal

```
podman build -t aml .
```

1. Run command in terminal (e.g. port 9000)

```
podman run -p 9000:80 --env-file env.list --  
name aml-app aml
```

```
[root@localhost ~]# curl -k https://127.0.0.1:443 --cert ./certs/openssl-1.1.1c/cert.pem --key ./certs/openssl-1.1.1c/key.pem -v -n -o /dev/null
[...]
[2024-08-10 08:45:18.722 +0000] [INFO] root: heavy lifting
[2024-08-10 08:45:18.722 +0000] [INFO] root: configuration is validated
[2024-08-10 08:45:18.722 +0000] [INFO] root: loading gp pre-processing
[2024-08-10 08:45:18.799 +0000] [INFO] root: GP pre-processed transformed data loaded
[2024-08-10 08:45:18.799 +0000] [INFO] root: Authentication token generation
[2024-08-10 08:45:18.799 +0000] [INFO] root: Authentication token generation
Unverified HTTPS request is being made to host "7.9.6.18:18". Adding certificate verification is strongly advised. See: https://urlﬁl.es/readthedocs.io/en/1.25.x/advanced-usage.html#ssl-warnings
[2024-08-10 08:45:18.813 +0000] [INFO] root: loading gp availability
[2024-08-10 08:45:18.943 +0000] [INFO] root: heavy lifting complete
```

---

1. AI model training

Get model details for inferencing

Configure sample application

Build sample application

Deploy sample application

[Access sample application](#)

2. AI model deployment

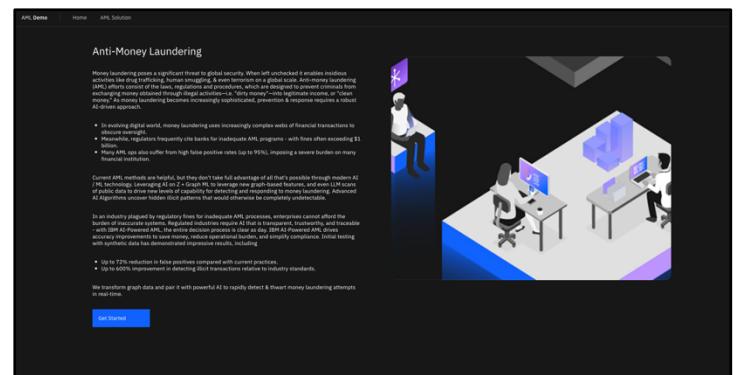
## Access sample application

1. View the following URL in a web browser

<http://{ip address}:{port}/ui/>

ip address: IP of server you deployed application in  
port: port you used with podman run

4. AI model integration



---

1. AI model training

2. AI model deployment

4. AI model integration

AI model integration complete