AI on IBM Z

Fraud detection 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 fraud detection solution on IBM Z.

IBM

Table of contents

Step 1

AI model training

We will build a fraud detection 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

```
ai-st-fraud-detection/zST-model-training-jupyter
```

Prerequisites

1. Must have Python (3.9 or 3.10) installed

Dataset guidance

Sample open source credit card transaction dataset can be found on Kaggle -

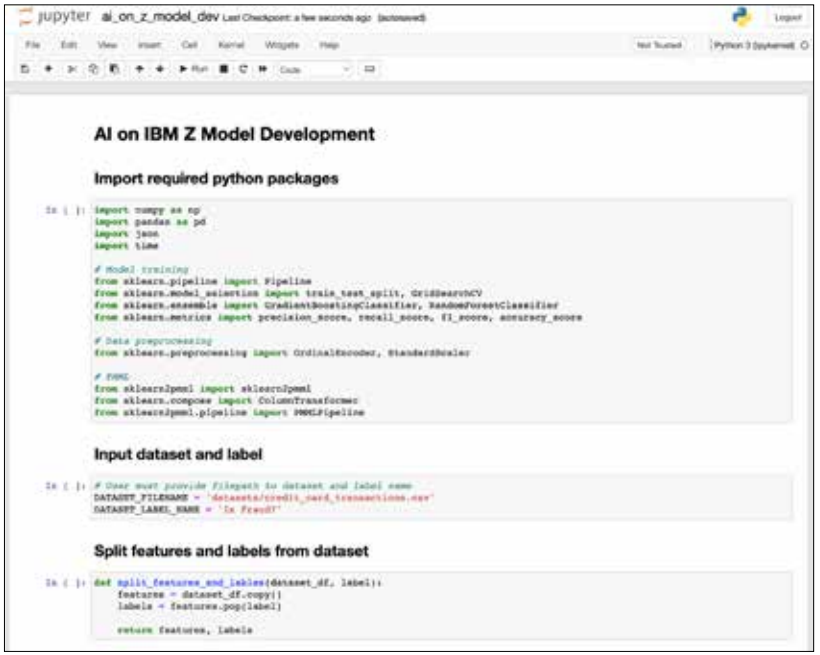
https://www.kaggle.com/datasets/ealtman2019/credit-card-transactions

There are several files included within the download. You can use *credit_card_transactions-ibm_v2.csv* for training. Due to the size of the sample dataset, the provided Jupyter notebook takes a subset of the data to decrease the training time. Please modifify the code in the “Fetch and process data” cell of the provided Jupyter notebook later to use more data during training.

Required features

- User (integer) – unique ID for user making transaction
- Card (integer) – unique ID for credit card
- Year (integer) – year of the transaction
- Month (integer) – month of the transaction
- Day (integer) – day of the month of the transaction
- Time (integer) - time of the transaction (HH:MM)
- Amount (float) – dollar amount of the transaction
- Use Chip (string) – the type of transaction (swipe transaction, etc)
- Merchant Name (integer) – unique ID for merchant name
- Zip (integer) – zip code of the transaction

Access rapid AI on IBM Z development environment



1. Create and activate Python virtual environment

```
python -m venv env
source env/bin/activate
```

2. Install required Python packages

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

3. Run Jupyter

```
jupyter notebook
```

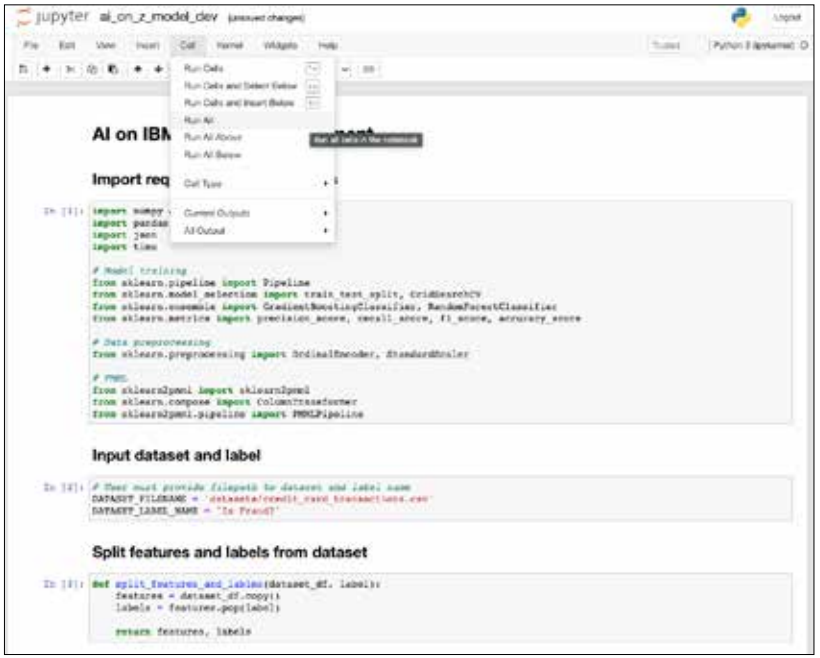
4. View Jupyter interface
   Go to localhost:8888 in a web browser

Provide data

1. Add your input dataset (csv) into datasets/ directory
2. Add input data to Jupyter notebook
   - Set DATASET_FILENAME to the path to your dataset
   - Set DATASET_LABEL_NAME to the name of the column you're predicting from the dataset

Model training

1. Step through and run Jupyter notebook from web browser



Access trained AI model

1. Once training is complete, you can find your AI models within the models/directory (choose one for the following AI model deployment step)

AI model training complete

Step 2

AI model deployment

We will deploy our fraud detection 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.

Prerequisites

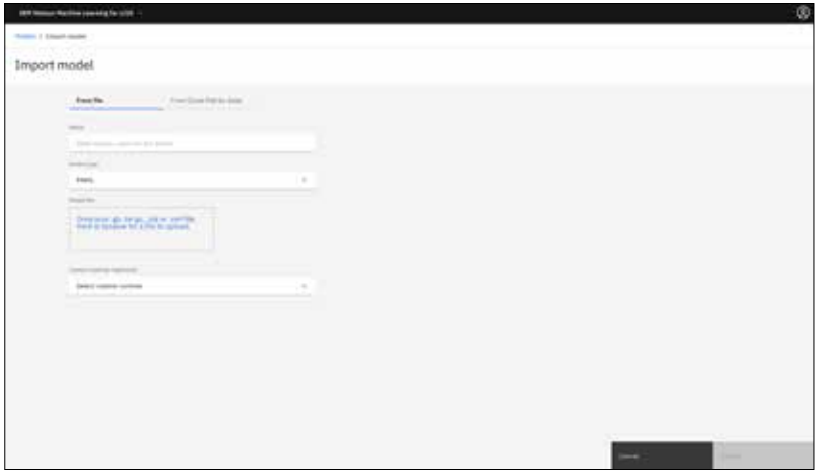
1. Must have MLz installed

Go to MLz UI

1. Sign in with username/password

Import AI model

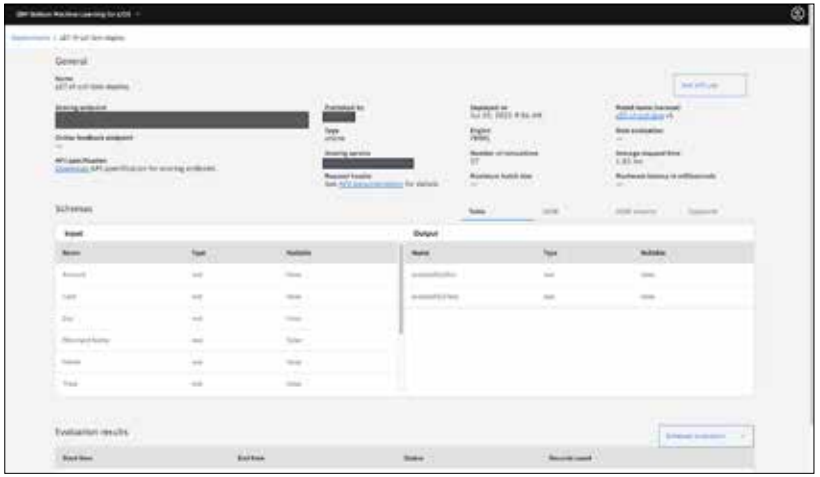
1. Go to models tab
2. Click import model
3. Enter model name
4. Choose model type
   Choose PMML if using your previously trained model
5. Drag and drop model file
   Use your previously trained model
6. Click import



Deploy AI model

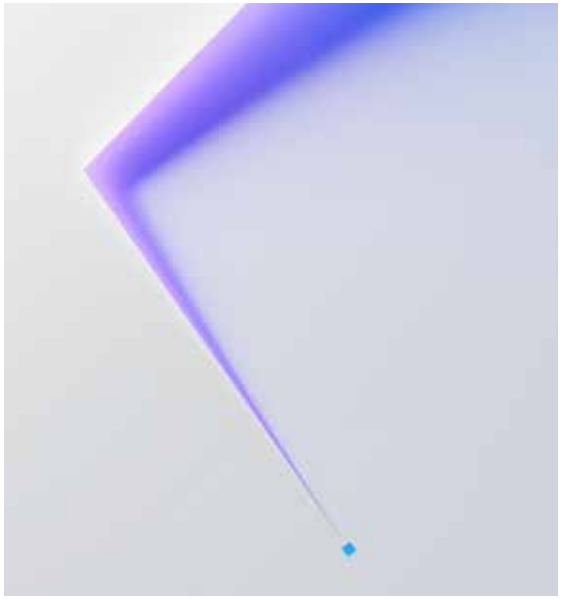
1. Go to models tab
2. Click action button for your model (on right side)
3. Click deploy
4. Enter deployment name
5. Choose deployment type
6. Choose model version
7. Choose scoring service
   Note: you should choose the correct scoring service based on your application (e.g. CICS or REST)
8. Click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

AI model deployment complete



Step 3

AI model analysis

We will analyze our fraud detection AI model using a sample AI on IBM Z Fraud Detection Dashboard. We can invoke the API of this sample dashboard from another sample application to visualize the AI model inferencing.

All sample code for this section is within

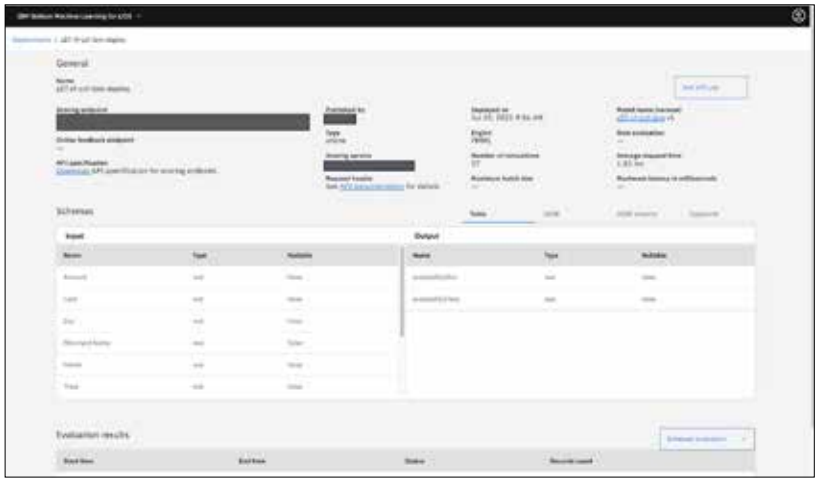
```
ai-st-fraud-detection/zST-model-analysis
```

Prerequisites

1. Must have MLz installed
2. Must have Python installed
3. Must have Git installed
4. Must have Docker or Podman installed

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



Configure sample application

1. Set the enrionment variables within

```
ai-st-fraud-detection/zST-model-analysis/
env.list
```

- WML_IP_W_PORT (IP address of MLz with port)
- WML_USER (MLz username)
- WML_PASS (MLz password)
- WMLZ_ENDPOINT (scoring endpoint for deployed AI model)

Build sample application

1. Run command in terminal

```
docker build -t model-analysis .
```

Deploy sample application

1. Run command in terminal (e.g. port 5002)

```
docker run --rm -p 5002:5002 --env-file
env.list --name model-analysis-app
model-analysis
```

Access sample application

1. View the following URL in a web browser
   http://{ip address}:{port}
   - IP address: IP of server you deployed application in
   - Port: port you used with Docker run

Analyze credit card events

1. Go to sample insights dashboard in web browser http://{ip address}:{port}
2. Go to latest predictions tab

Make predication

1. Go to sample insights dashboard in web browser http://{ip address}:{port}
2. Go to latest make predication tab
3. Input json data
4. Click submit

AI model analysis complete

Step 4

AI model integration

Choose One:

Ecommerce web application
CICS-COBOL application

We can use our deployed MLz fraud detection AI model and integrate it into different types of applications. The AI model can be analyzed and/or provide inferencing APIs using the sample AI on IBM Z Fraud Detection Dashboard.

Ecommerce web application

The sample e-commerce application is based on the open source EverShop Storefront and has been extended to integrate with the AI on IBM Z Solution Template. EverShop is a GraphQL Based and React ecommerce platform with essential commerce features. Built with React, modular and fully customizable.

All sample code for this section is within

```
ai-st-fraud-detection/zST-storefront-evershop
```

Prerequisuties

1. Must have AI on IBM Z Sample Fraud Detection Dashboard deployed for inferencing and analysis
2. Must have Docker installed

Install and deploy sample ecommerce application

1. Set app_url_w_port variable in CheckoutForm.jsx to your server IP and port (ip:port)

```
ai-st-fraud-detection/zST-storefront-
evershop/packages/evershop/src/components/
frontStore/stripe/checkout/CheckoutForm.
jsx
```

2. Run command in terminal

```
docker-compose up
```

Configure sample ecommerce application

Access admin panel from web browser

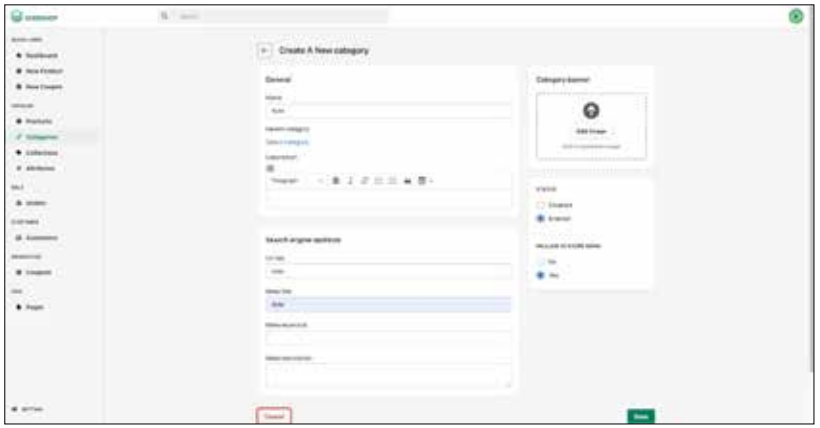
1. Enter URL in web browser using app url (e.g. localhost)

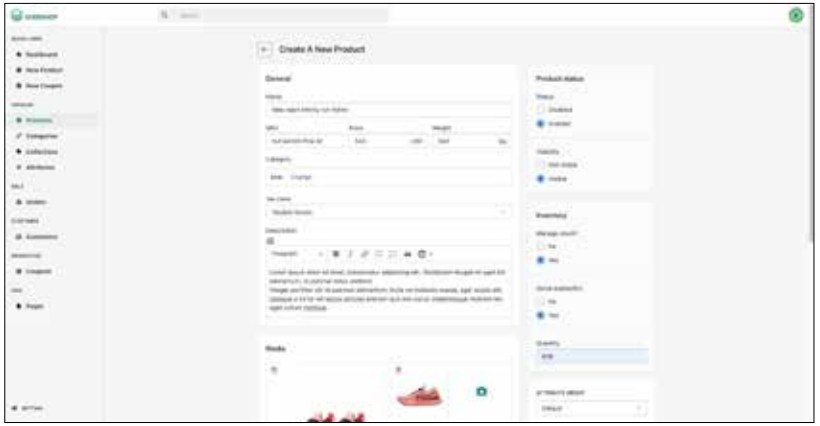
```
http://localhost:3000/admin
```

2. Login with default admin credentials
   - Email: admin@test.com
   - Password: password

Add products

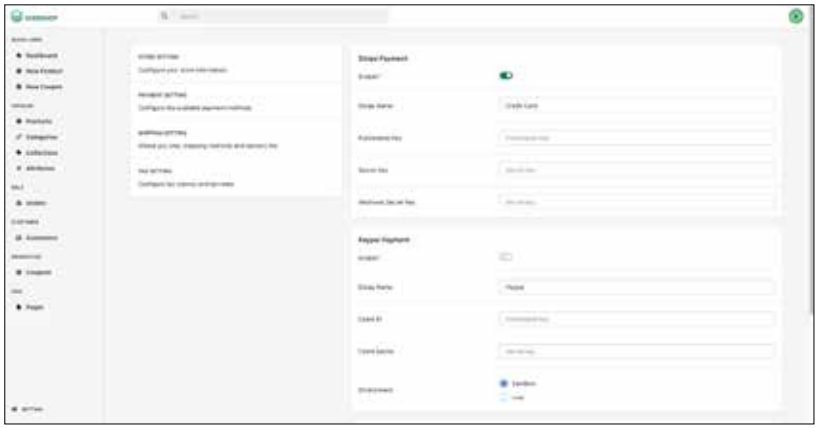
1. Create categories
   - Click categories from the catalog section
   - Click new category
   - Add category details
     - Add name (e.g. Kids)
     - Add url key
     - Add meta title
     - Change status to enabled
     - Change include in store menu to yes
   - Click save
2. Add products
   - Click products from the catalog section
   - Click new product
   - Add category details
     - Make sure to change add category
     - Make sure to change status to enabled
     - Make sure to change visibility to visible





Configure store settings

1. Enter URL in web browser using app url (e.g. localhost)
2. Click store setting

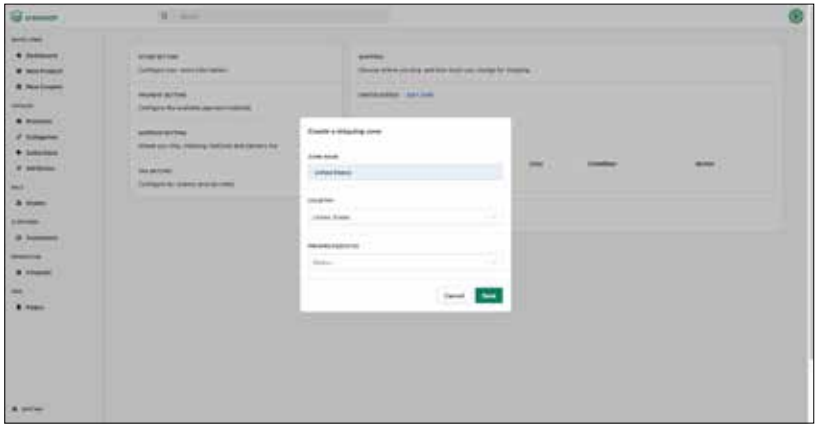


Configure payment settings

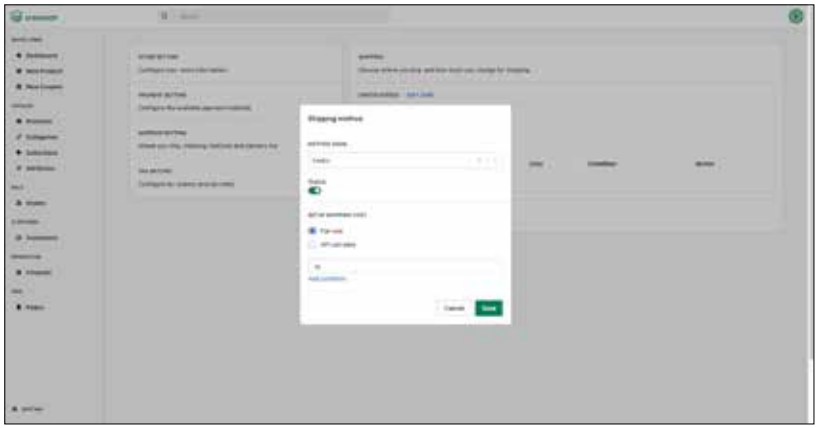
1. Click setting on the bottom left
2. Click payment setting
3. Enable stripe payment
4. Click save

Configure shipping settings

1. Add shipping zone
   - Click setting on the bottom left
   - Click shipping setting
   - Click create new shipping zone
   - Add shipping details
   - Click save



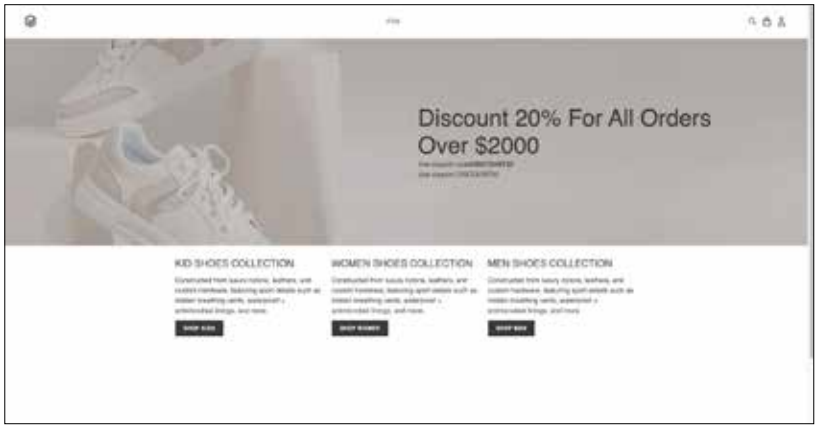
2. Add payment method
   - Click add method
   - Add method name (e.g. FedEx)
   - Enable status
   - Add flat rate (e.g. 10)
   - Click save



Access sampe ecommerce application

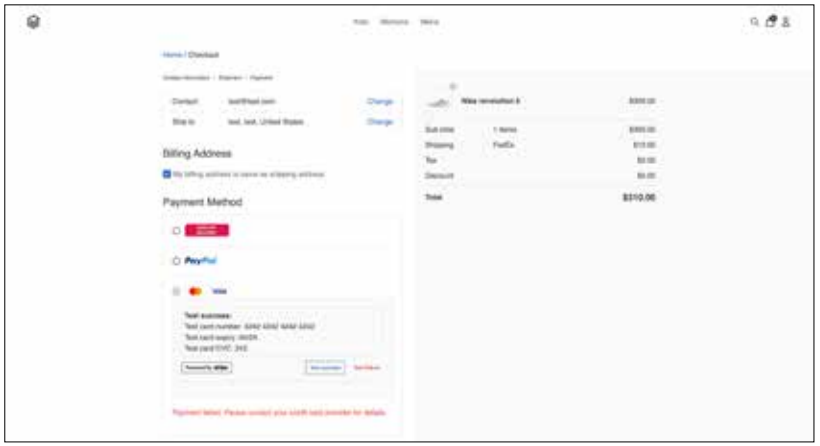
1. Enter URL in web browser using app url (e.g. localhost)

```
http://localhost:3000
```



Use fraud detection AI model with EverShop Storefront

1. Make sure you have AI on IBM Z Sample Fraud Detection Dashboard deployed for inferencing and analysis on the same local system
2. AI on IBM Z Sample Fraud Detection Dashboard is configured to invoke MLz AI model
3. Add items to cart
4. Place order
   - Choose test failure as payment method for fraud transaction example
   - Choose test success as payment method for non fraud transaction example



AI model integration complete

CICS-COBOL application

We can use our deployed MLz fraud detection AI model and integrate it into different types of applications. Guidance on integrating the AI model into a sample CICS-COBOL application is below.

All sample code for this section is within

```
ai-st-fraud-detection/zST-model-integration-CICS
```

Prerequisuties

1. Must have access to z/OS CICS environment
2. Must have MLz installed
3. Must have model deployed with the CICS-scoring server as scoring service
4. Must have the JVM server and bundle setup and running. To verify the same execute the below commands in the CICS.

   JMVSERVER

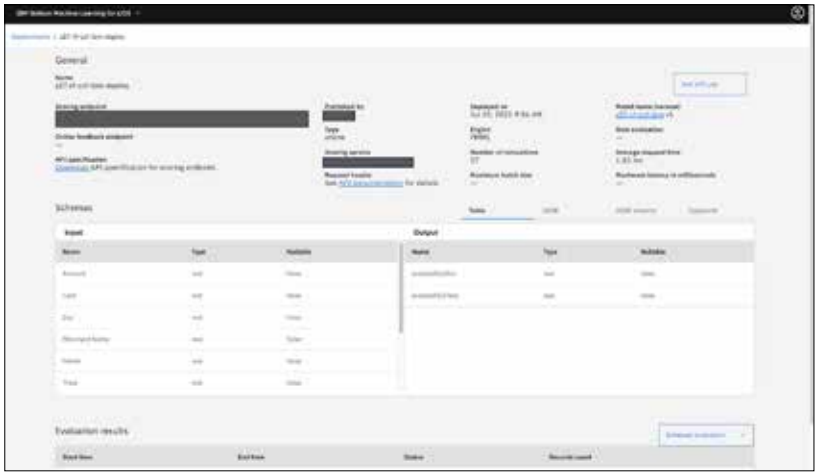
   ```
   CEMT INQUIRE JVMSERVER(ALNSCSER)
   ```

   BUNDLE

   ```
   CEMT INQUIRE BUNDLE(ALNSCBDL)
   ```

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



Integrate into CICS application

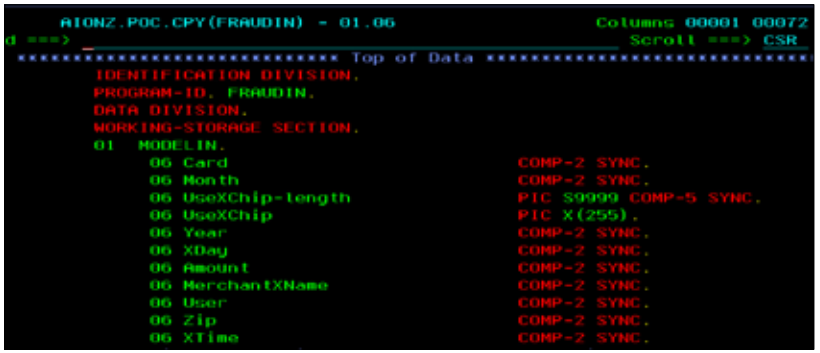
1. Copy the input and output copybooks from MLz UI and paste them into two different files on z/OS. Sample input and output files are provided as an example
   - Input copybook member
     - Create FRAUDIN file
     - Copy input copybook from MLz UI

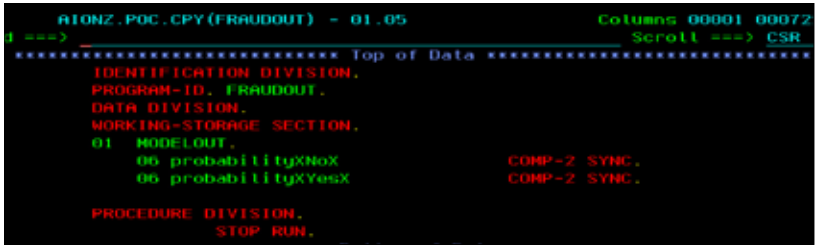
       ```
       ai-st-fraud-detection/zST-model-integration-CICS/FRAUDIN.cpy
       ```

     - Paste input copybook into FRAUDIN file
     - Save file
   - Output copybook member
     - Create FRAUDIN file
     - Copy output copybook from MLz UI

       ```
       ai-st-fraud-detection/zST-model-integration-CICS/FRAUDOUT.cpy
       ```

     - Paste output copybook into FRAUDOUT file
     - Save file





Get model details for inferencing

Integrate into CICS Application

2. Generate input and output Java helper classes by executing the ALNJCGEN JCL individually. Sample ALNJGEN JCL is provided as an example

   - Create ALNJCGEN JCL
     - Create a file for ALNJCGEN Job
     - Copy provided sample

       ```
       ai-st-fraud-detection/zST-model-
       integration-CICS/ALNJCGEN.jcl
       ```

     - Paste sample ALNJCGEN into ALNJCGEN file
   - Generate input Java helper class
     - Update ALNJCGEN
       - Update the JCL to give the copybook as FRAUDOUT
       - Update the path of the copybook accordingly
       - Give appropriate name for the output Wrapper class. PMMLOutPipeWrapperFraud
     - Submit job
     - Verify job ran successfully (MAXCC = 0)
   - Generated Wrapper classes can be found in the path

     ```
     /wmlhome/cics-scoring/usr/SCCSA01/
     servers/ALNSCSER/lib/global/com/ibm/
     cicsdev/bean
     ```

Get model details for inferencing

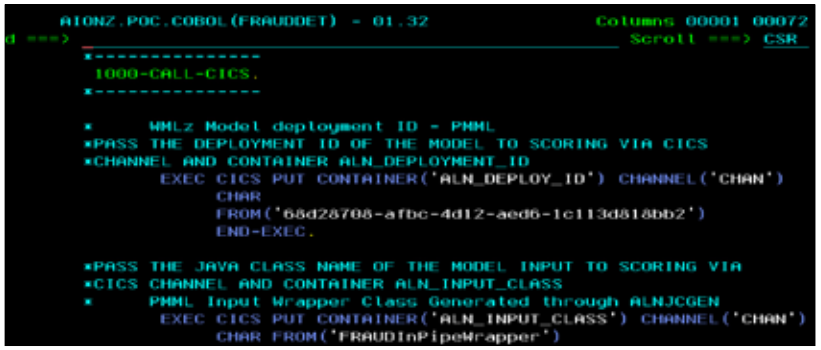
Integrate into CICS Application

3. Create COBOL program

- Create ALNJCGEN JCL
- Copy provided sample COBOL program

```
ai-st-fraud-detection/zST-model-integration-CICS/FRAUDDET.cob
```

- Paste sample COBOL program into FRAUDDET
- Copy deployment ID from MLz UI
- Replace DEPLOYMENT_ID in sample COBOL program with your deployment ID
- Update the names of the input and output wrapper classes created in the previous

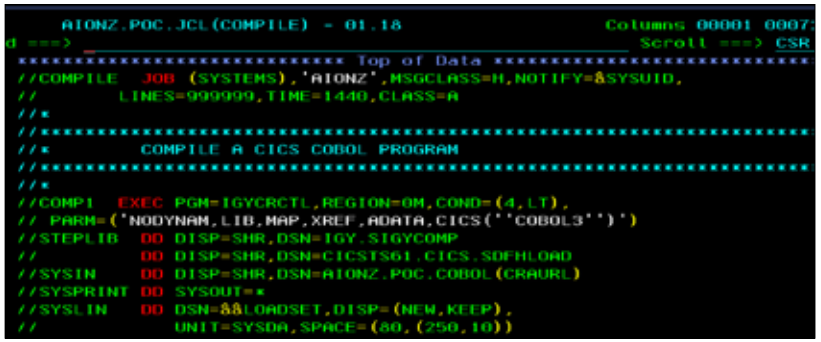


4. Compile and link-edit CICS-COBOL program

- Create COMPILE file
- Copy provided sample COMPILE file

```
ai-st-fraud-detection/zST-model-integration-CICS/COMPILE.jcl
```

- Paste sample COMPILE file into COMPILE file
- Paste sample COMPILE file into COMPILE file
- Submit job
- Verify job ran successfully (MAXCC = 0)



Get model details for inferencing

Integrate into CICS Application

5. Define the transaction and program in the CICS region

   - Transaction:

     ```
     CEDA DEFINE TRANS(FRDT) GROUP(ALNSCGRP)
     PROGRAM(FRAUDDET) DESCRIPTION(TRAN to
     execute FRAUDDET)
     ```

   - Program:

     ```
     CEDA DEFINE PROGRAM(FRAUDDET)
     GROUP(ALNSCGRP) LANGUAGE(COBOL)
     DESCRIPTION(FRAUD DETECTION)
     ```

6. Install transaction and program in the CICS region

   - Transaction:

     ```
     CEDA INS TRANS(FRDT) GROUP(ALNSCGRP)
     ```

   - Program:

     ```
     CEDA INS PROGRAM(FRAUDDET) GROUP
     (ALNSCGRP)
     ```

7. Run transaction

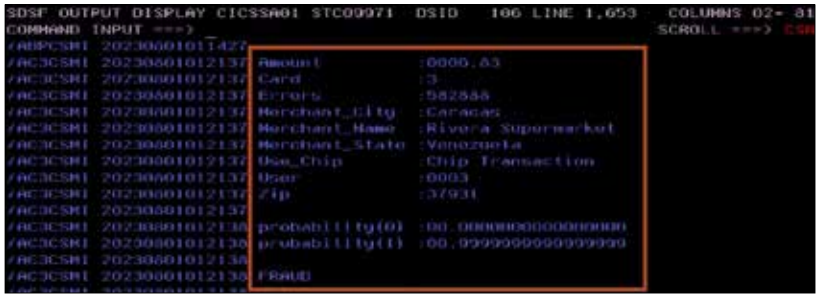
   ```
   FRDT
   ```

Get model details for inferencing

Integrate into CICS Application

8. View output results

   - Go to the Spool of the started task for CICS (e.g. CICSSA01)
   - Navigate to the CEEMSG dataset
   - View displays from the module in this dataset
     Sample output is shown as below



AI model integration complete