AI on IBM Z

Fraud detection solution template

This solution template provides an example on how to deploy AI using a Linux on Z environment, while making use of open source frameworks, Triton Inference Server (TIS), 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.

Abstract

This solution demonstrates how to train, evaluate, and deploy fraud detection models on Linux on IBM Z, leveraging machine learning and containerized inferencing. The workflow begins with data acquisition, where users download transaction datasets from Kaggle. These datasets contain various transaction features such as Transaction Amount, Merchant Name, Use Chip, and Zip Code, along with a label indicating whether the transaction was fraudulent (1) or legitimate (0). The labeled dataset enables the use of supervised learning techniques to train models that can identify fraud patterns.

Once the dataset is prepared, users train multiple machine learning models using Scikit-learn within a Jupyter Lab environment. The training process involves data preprocessing, including handling missing values, removing special characters, and categorizing features as numerical or categorical. The pipeline then trains and evaluates multiple models, such as Random Forest and Gradient Boosting, applying hyperparameter tuning to optimize performance. The models are ranked based on accuracy, precision, recall, and F1-score, ensuring that the most effective model is identified for deployment.

After training, the best-performing model is exported and deployed to Triton Inference Server, which runs inside a Podman container. Triton enables efficient, scalable AI inferencing, allowing real-time fraud detection. While Podman is used in this demo for simplicity, real-world deployments would typically use IBM Machine Learning for Z (MLz) or IBM Cloud Pak for Data (CP4D) to provide a more robust and scalable infrastructure.

The final stage of the solution integrates fraud detection into an e-commerce application, which is also deployed in Podman. When a customer proceeds to checkout, the application interacts with the Triton inference server, sending transaction details for fraud evaluation. The model predicts whether the transaction is fraudulent, allowing the application to take appropriate action, such as approving, flagging, or blocking the transaction. This demonstrates how AI-powered fraud detection can be seamlessly embedded into business applications, ensuring security and reducing financial risks.

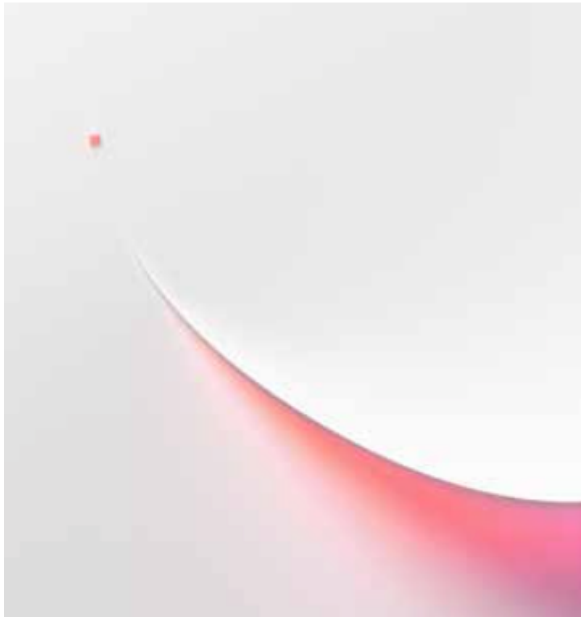
Table of contents

AI model training.....3

AI model deployment.....8

AI model analysis.....12

AI model integration.....17



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

C:\Users\Administrator\Documents\ai-solution-templates

Environments used

Windows

Dataset guidance

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

C:\Users\Administrator\Documents\ai-solution-templates\datasets\credit_card_transactions-ibm_v2.csv

Due to the size of the sample dataset, the provided Jupyter notebook takes a subset of the data to decrease the training time. Please modify the code in the “Fetch and process data” cell of the provided Jupyter notebook later to use more data during training.

It can also be found on Kaggle -

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

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
- Merchant Name (integer) – unique ID for merchant name
- Zip (integer) – zip code of the transaction

Access rapid AI on IBM Z development environment

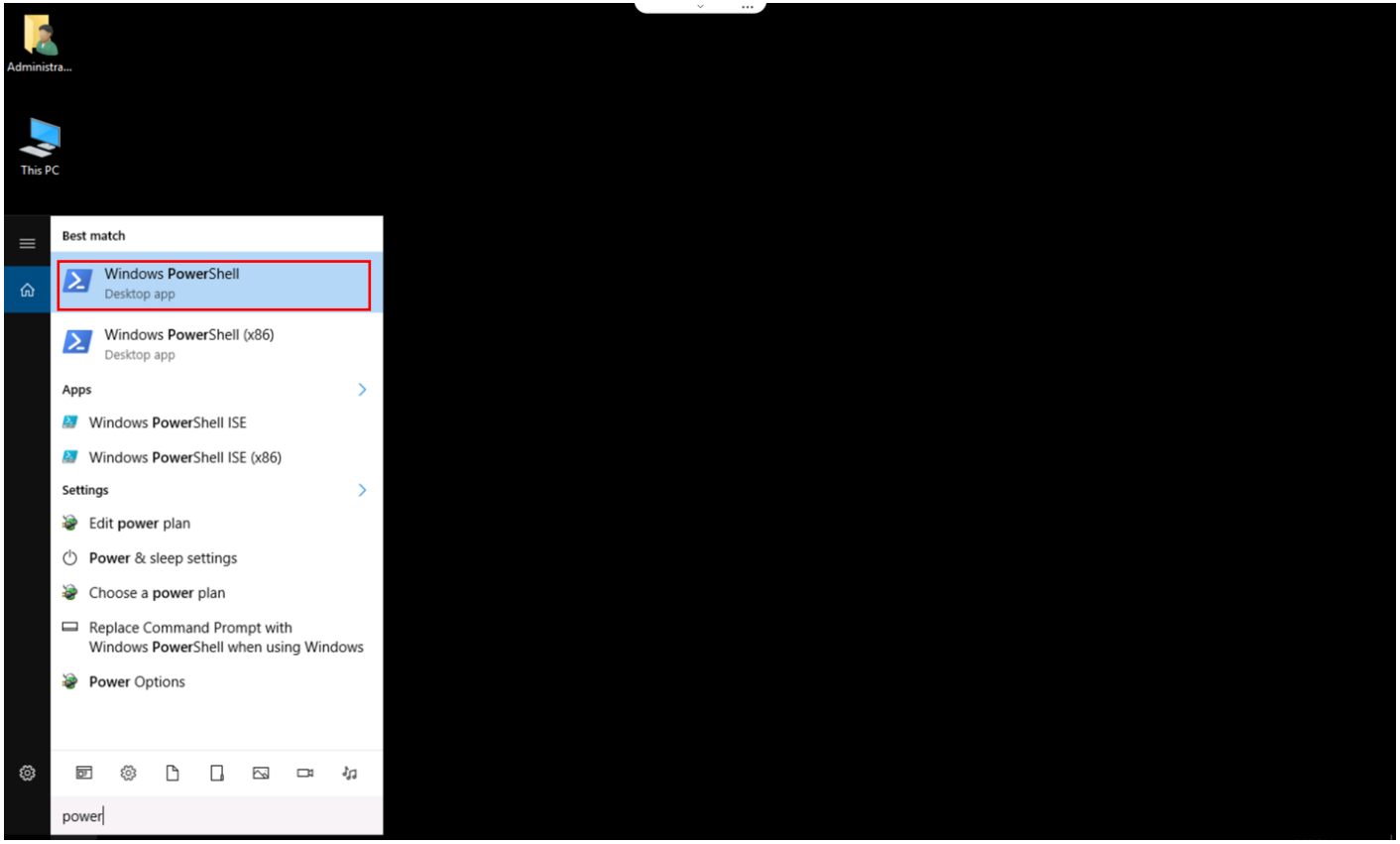
Provide data

Model training

Access trained AI model

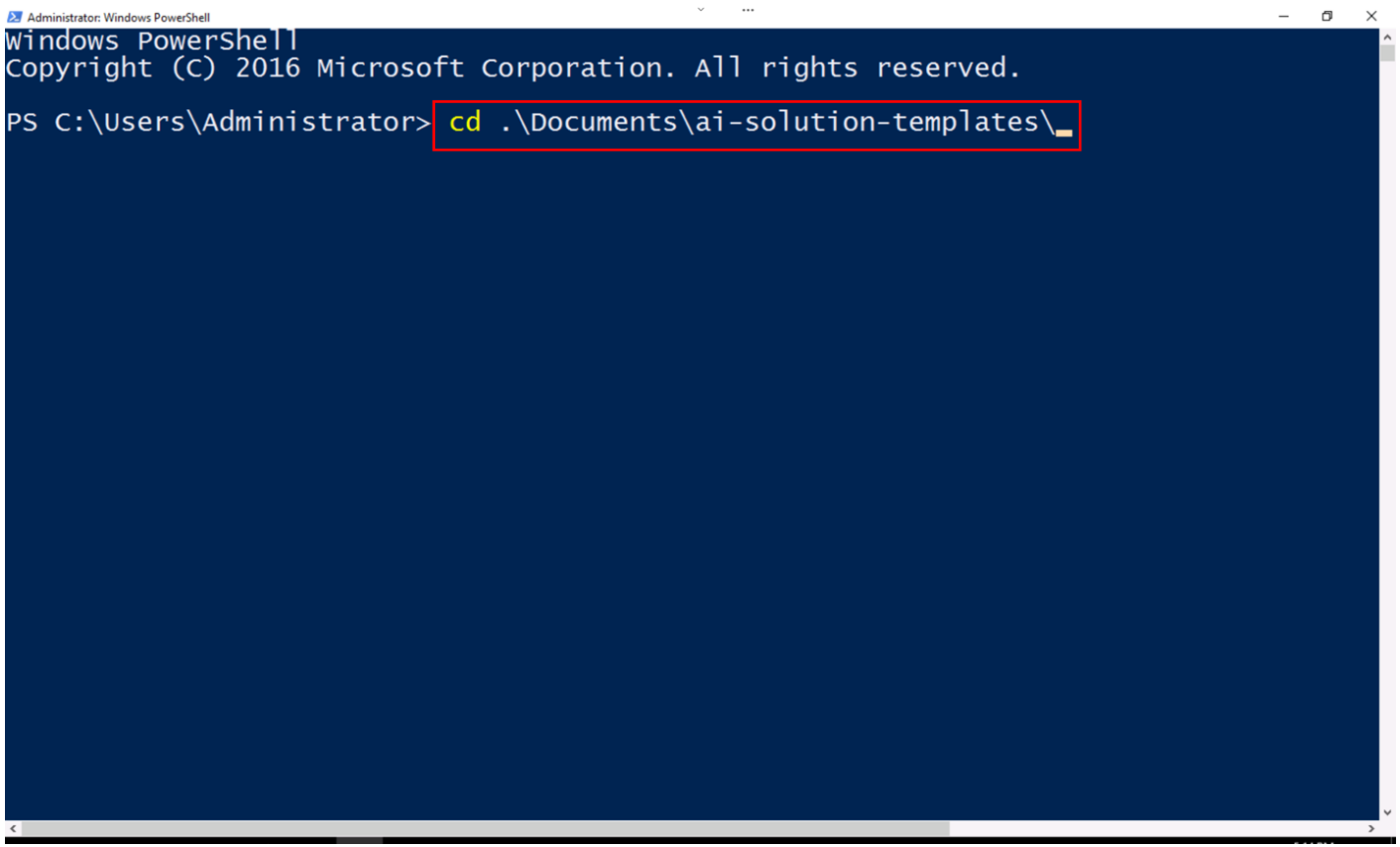
Access rapid AI on IBM Z development environment

1. Open Windows PowerShell



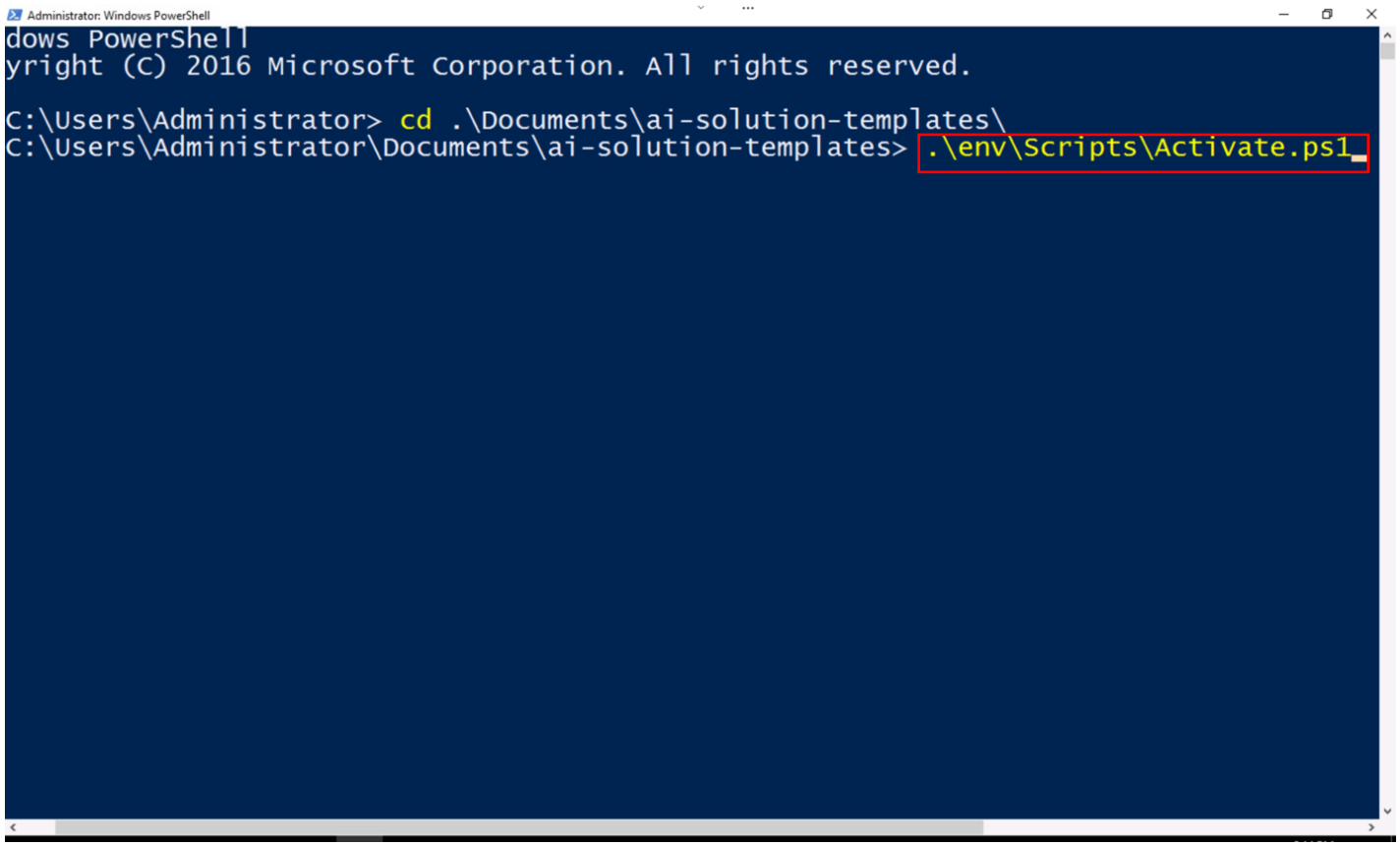
2. Go to model training pipeline

   `cd .\Documents\ai-solution-templates\`



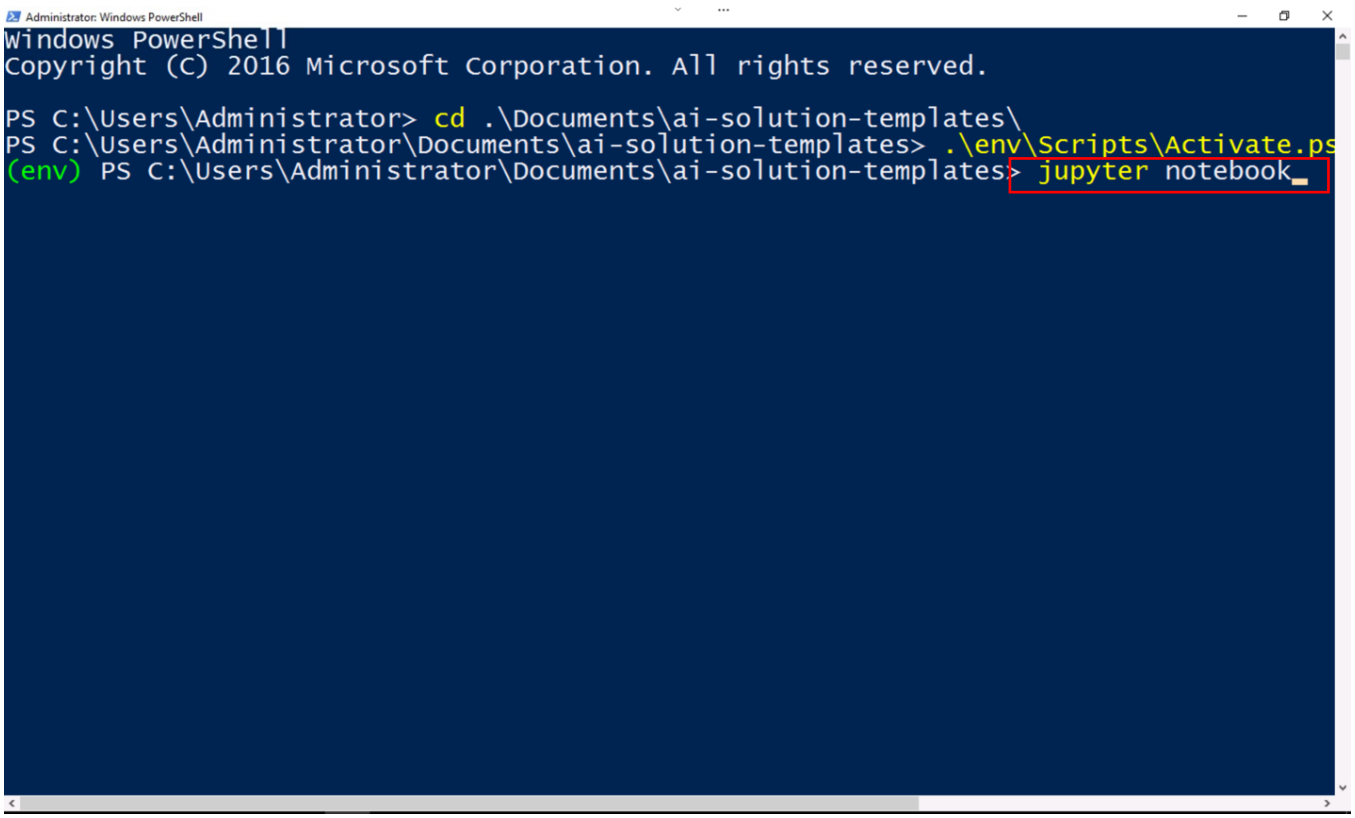
3. Activate Python virtual environment

   `.\env\Scripts\Activate.ps1`

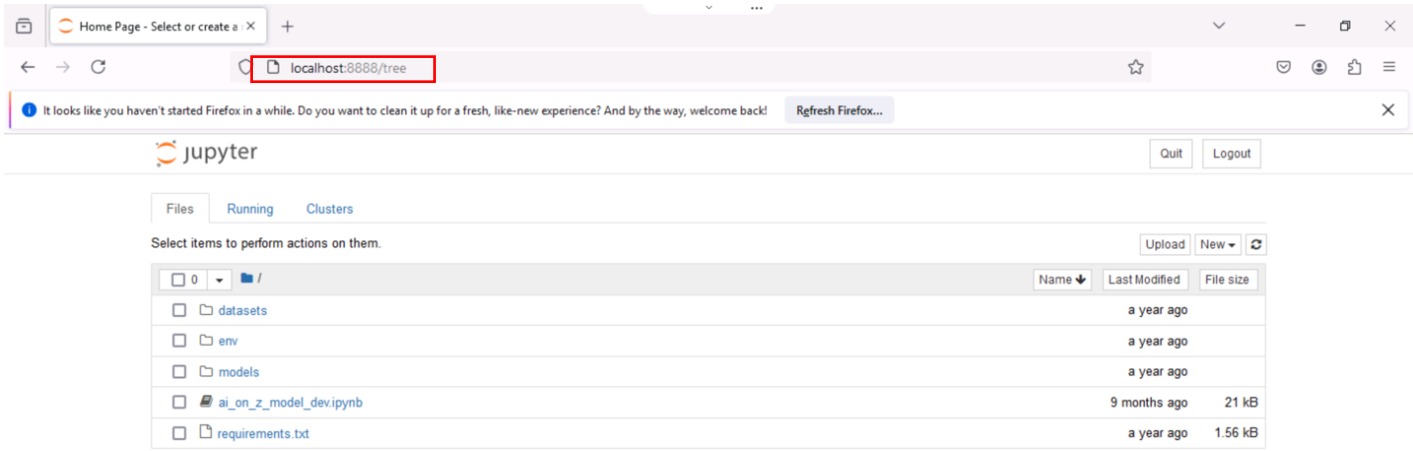


4. Run Jupyter

   jupyter notebook



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



Access rapid AI on IBM Z development environment

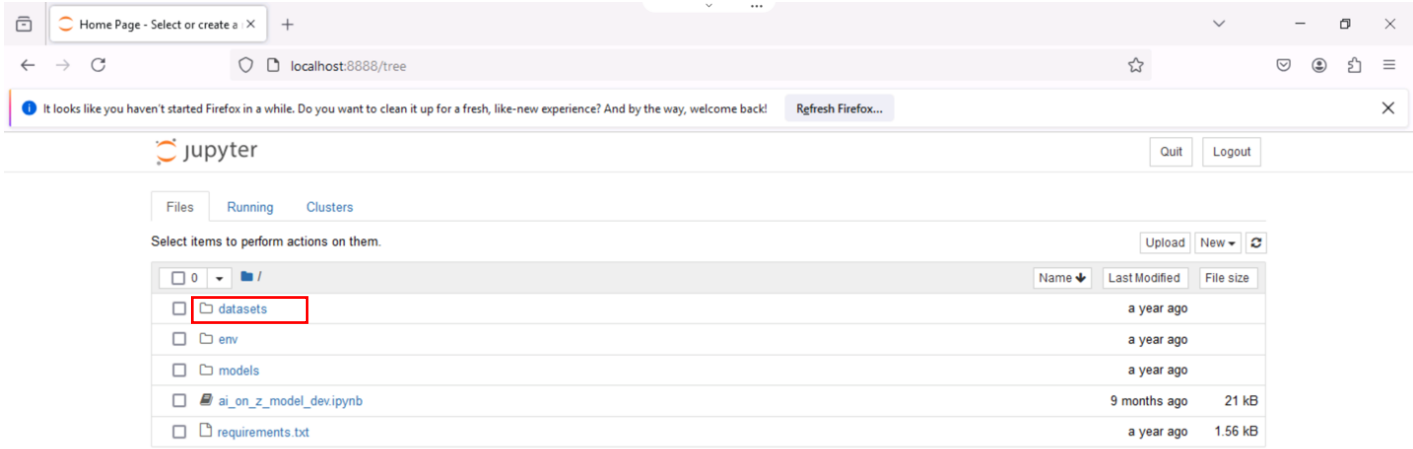
Provide data

Model training

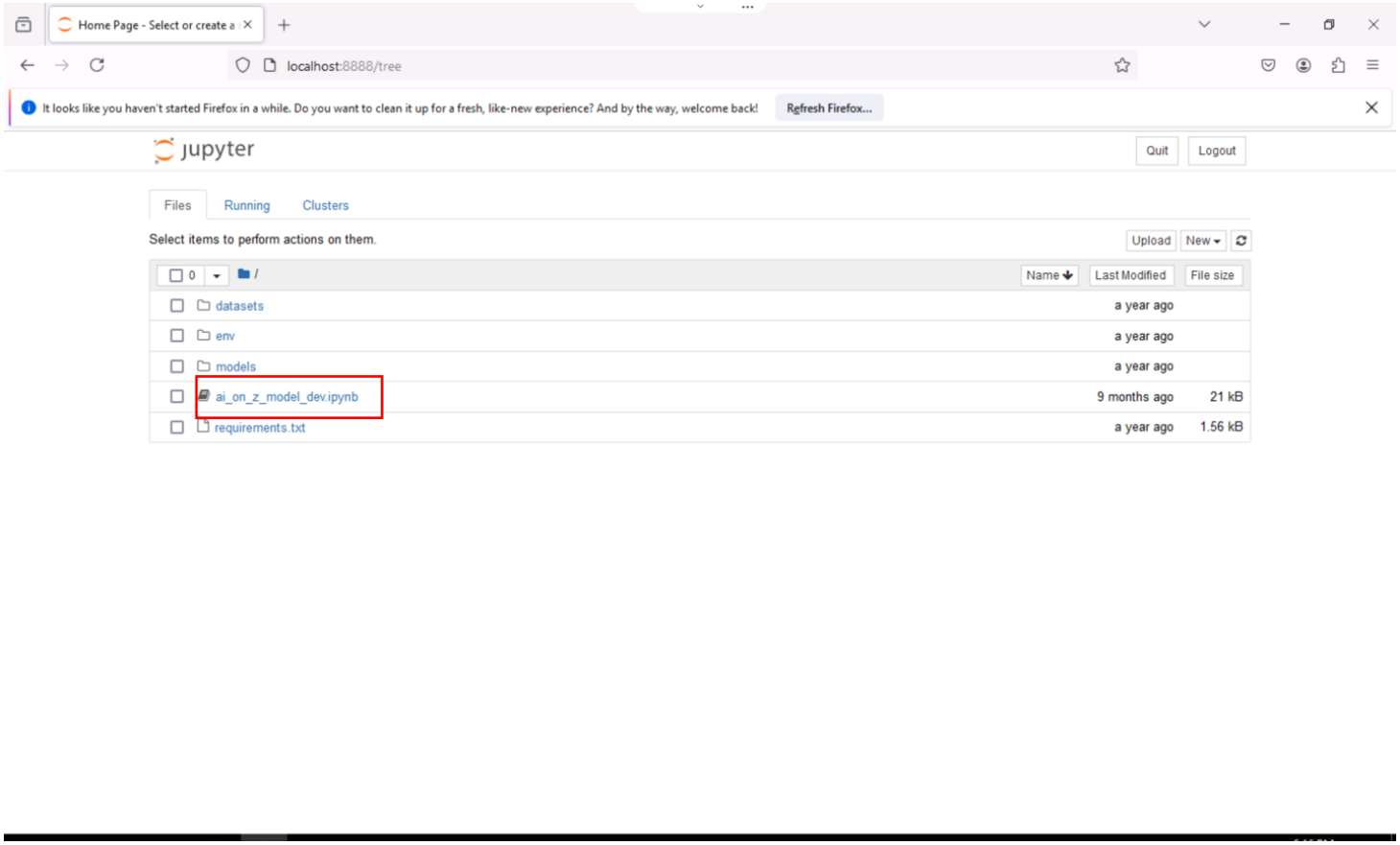
Access trained AI model

Provide data (optional)

1. Your input dataset (csv) in datasets/ directory

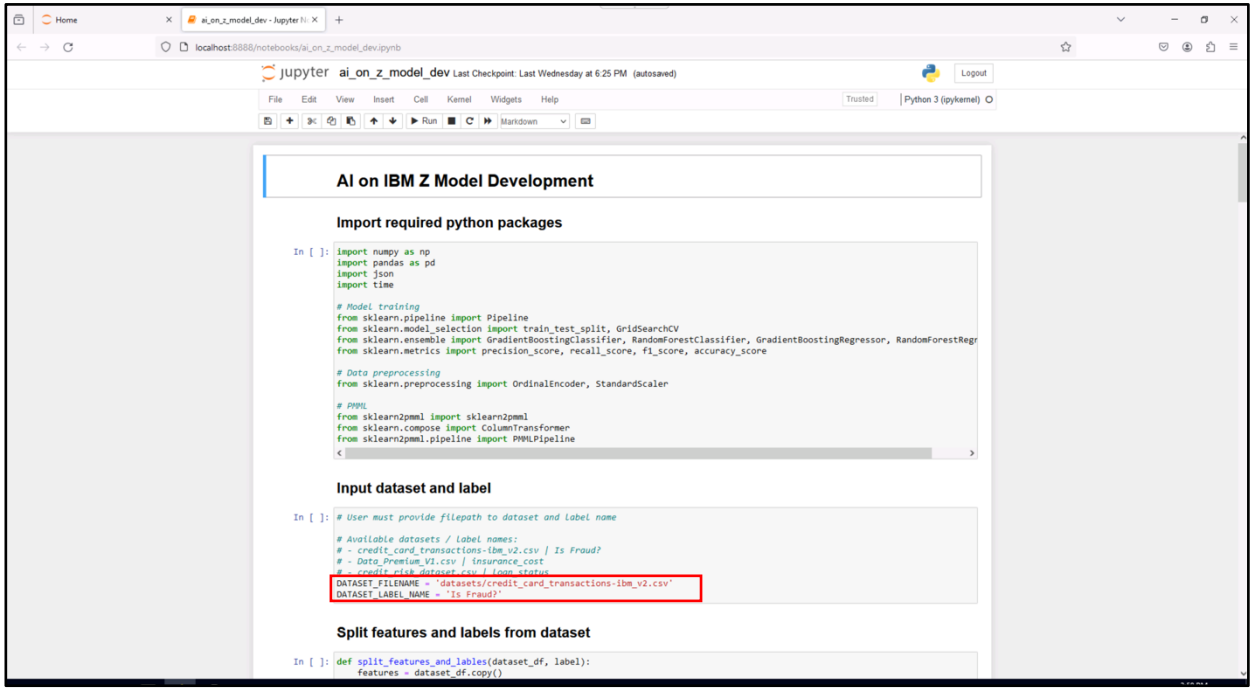


2. Add input data to Jupyter notebook (ai_on_z_model_dev.ipynb)



a. Set `DATASET_FILENAME` to the path to your dataset

b. Set `DATASET_LABEL_NAME` to the name of the column you're predicting from the dataset



Model training

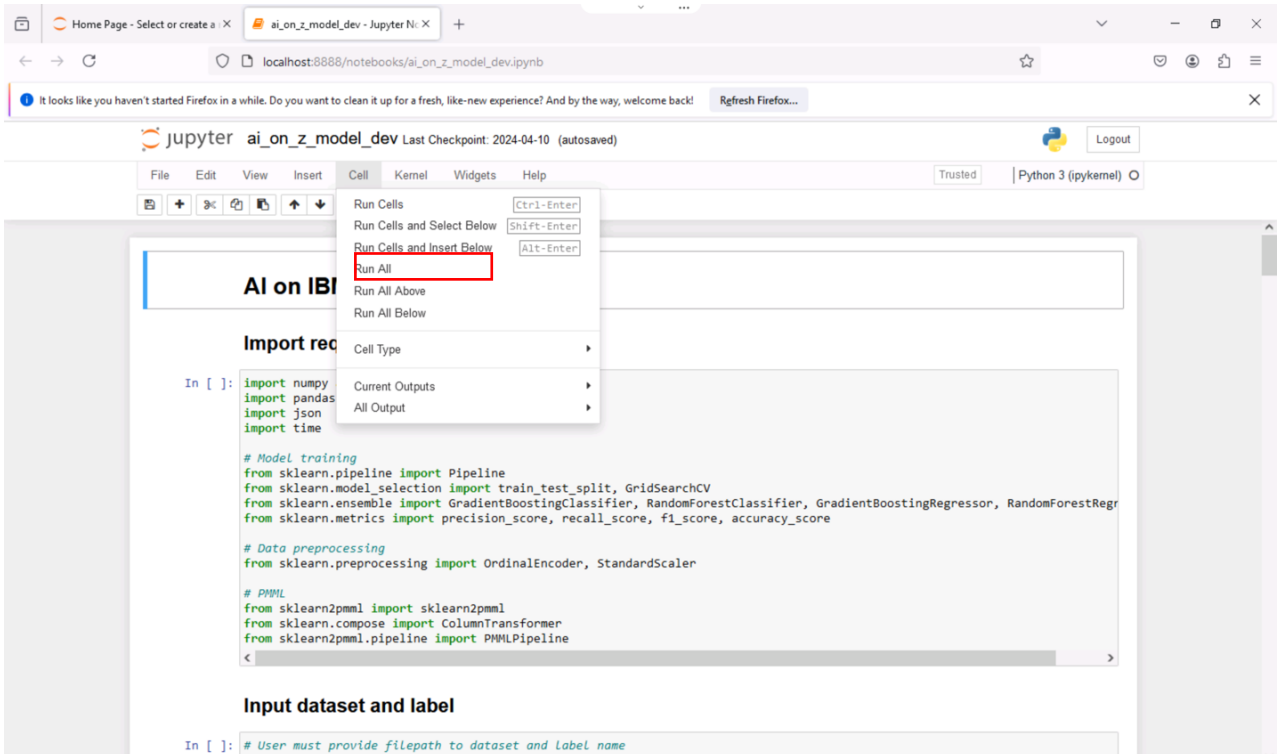
1. Step through and run all cells within Jupyter notebook (ai_on_z_model_dev.ipynb) within web browser

This pipeline automates the entire machine learning workflow, from data ingestion to model selection and evaluation. It begins by loading and validating the dataset, ensuring it contains the necessary features for a supported AI use case, such as fraud detection. The data is then cleaned and preprocessed, handling missing values, removing unnecessary characters, and categorizing features as numerical or categorical.

Next, the pipeline determines whether the problem is classification or regression based on the target variable. It then trains multiple machine learning models, applying feature transformations and hyperparameter tuning using Scikit-learn pipelines and GridSearchCV. For fraud detection, the models learn patterns from historical transaction data, identifying key fraud indicators such as unusual transaction amounts, location mismatches, and non-chip usage.

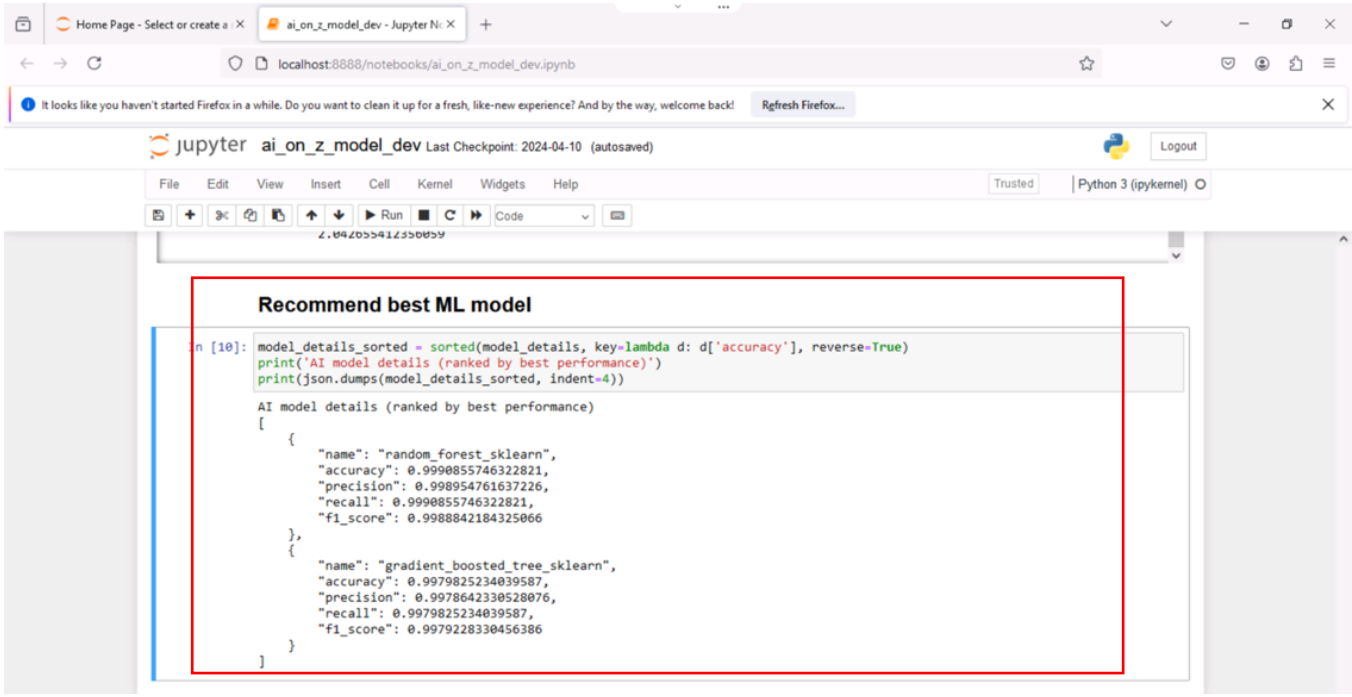
Note: This may take several minutes

1. AI model training. 2. AI model deployment 3. AI model analysis 4. AI model integration



The trained models are then evaluated based on accuracy, precision, recall, and F1-score to determine their effectiveness in distinguishing fraudulent transactions from legitimate ones.

Finally, the trained models are ranked by accuracy, allowing the user to identify the best-performing model. This ensures that only the most accurate model is selected for deployment in a production environment, such as an inference server like Triton, where it is used for real-time fraud detection. By automating this process, the pipeline ensures that organizations can efficiently train, evaluate, and deploy AI models to enhance fraud detection and decision-making.



Access rapid AI on IBM Z development environment

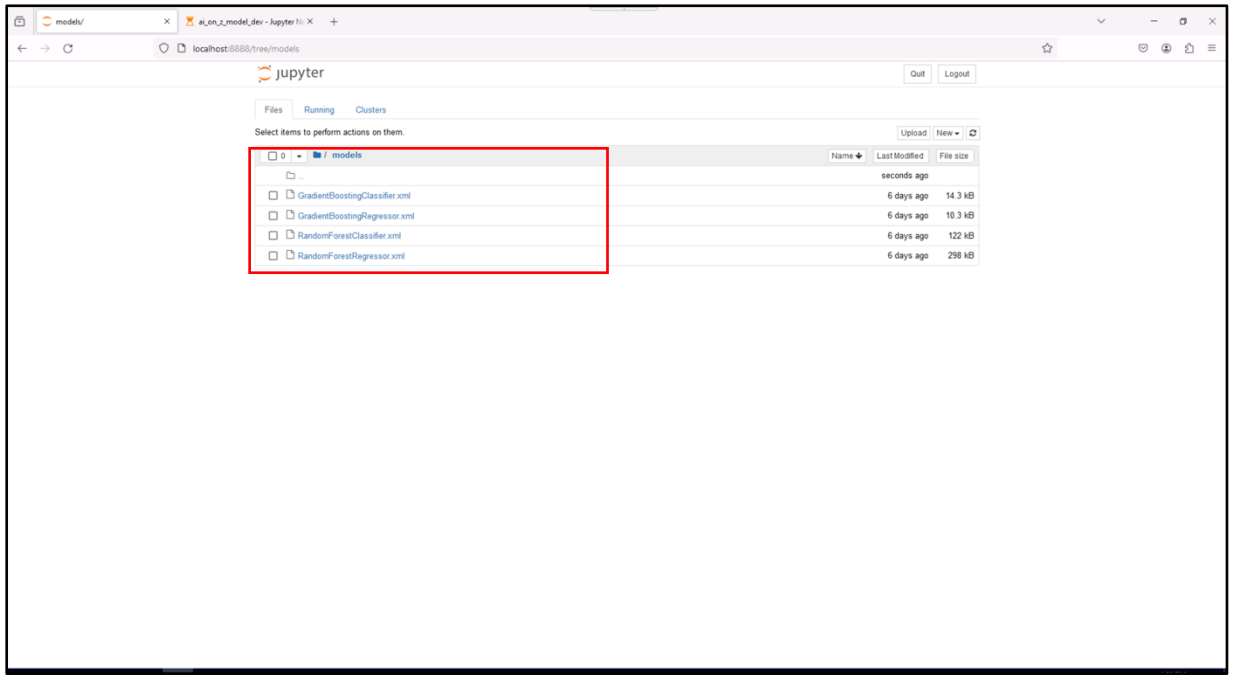
Provide data

Model training

Access trained AI model

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

Environments used

Windows
Linux on Z

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.

Access sample code

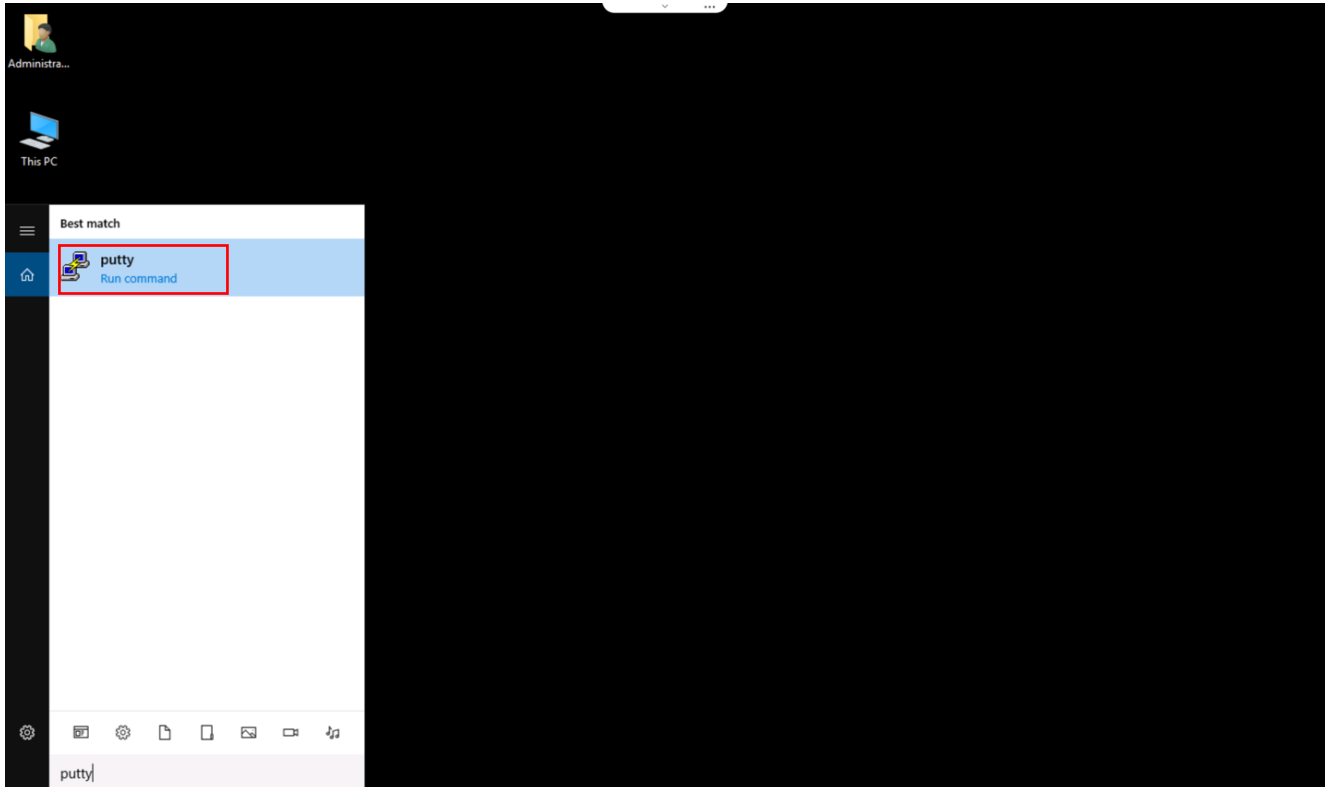
Integrate AI model into Triton Inference Server

Deploy Triton Inference Server

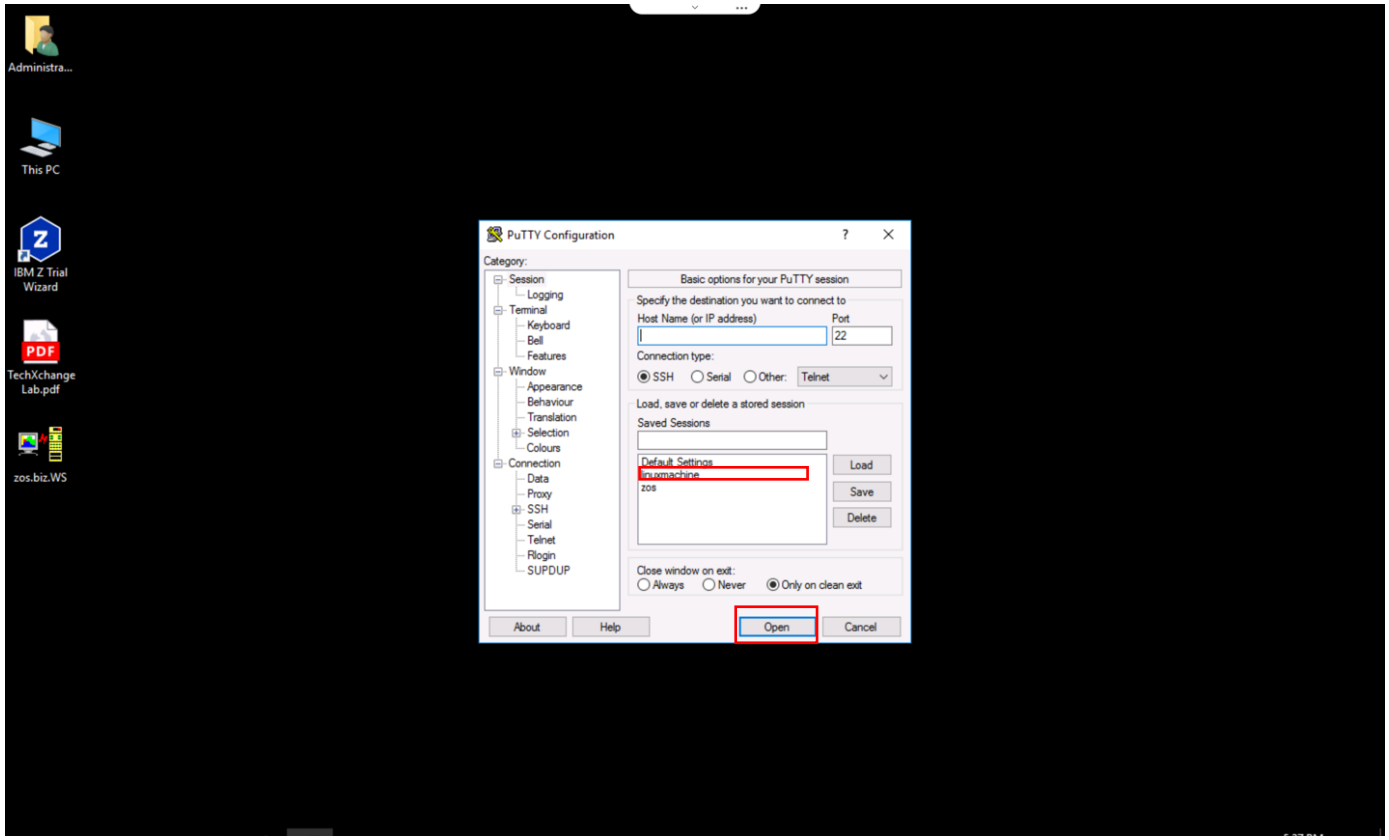
Run sample test

Access sample code

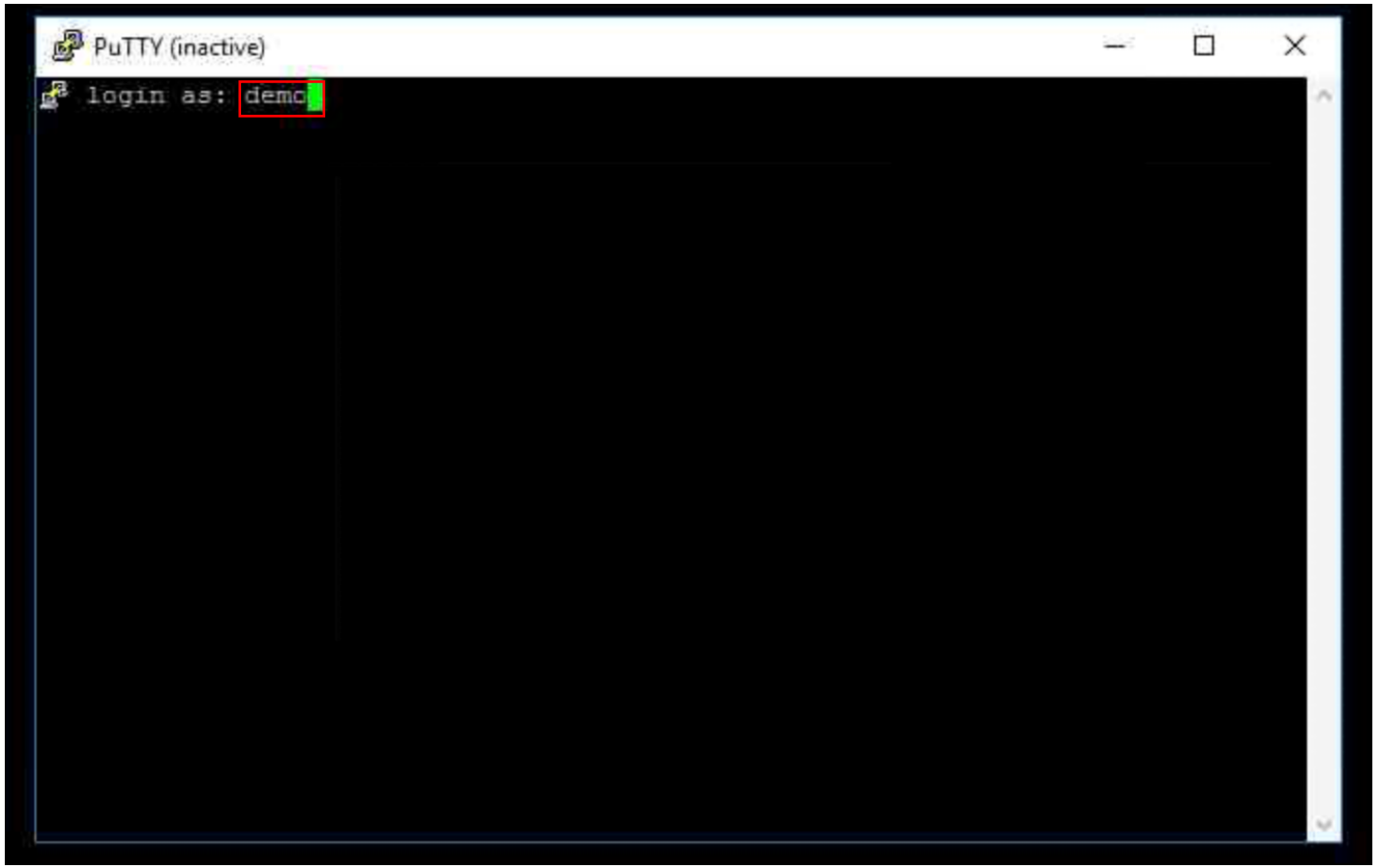
1. Open PuTTY application



2. Double click on linuxmachine

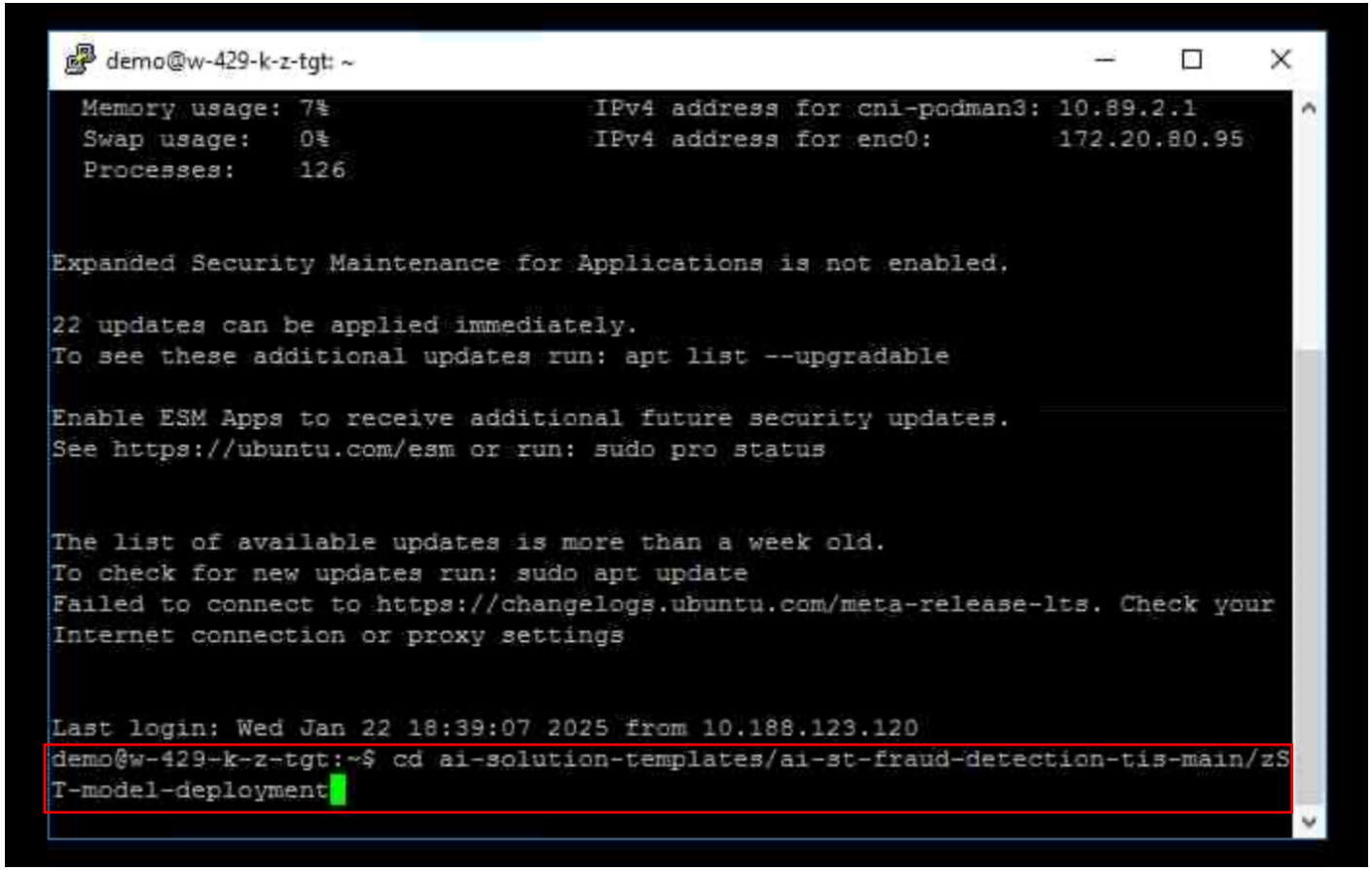


3. Enter demo username



4. Move to sample code directory

```
cd ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-deployment
```



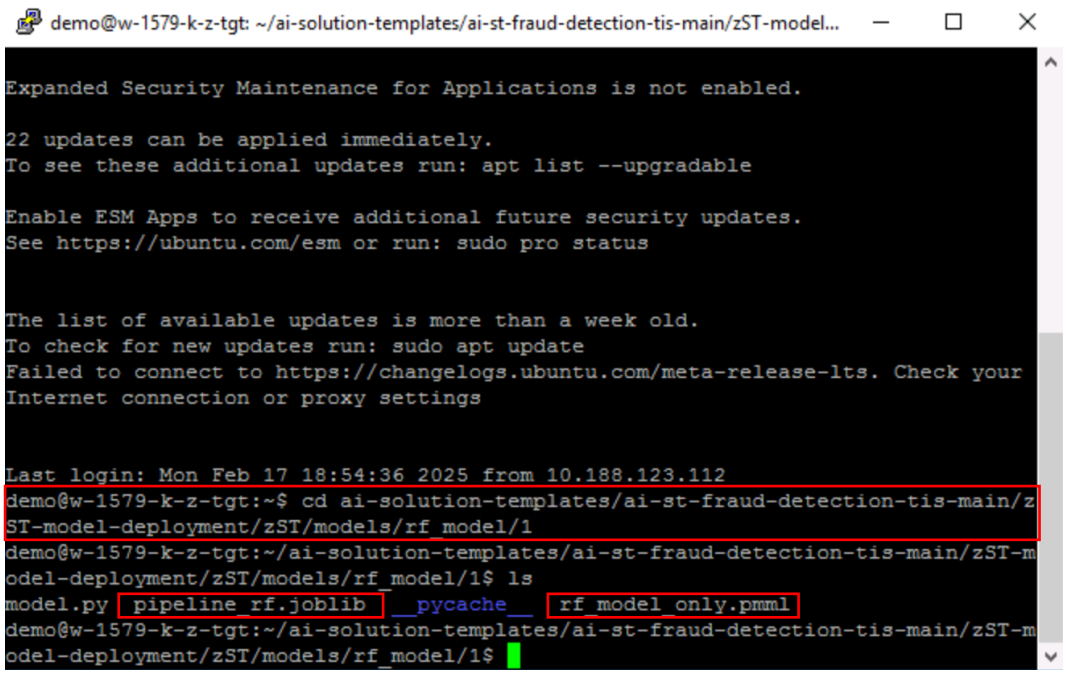
Integrate AI model into Triton Inference Server

1. Add your model (.pmml file) to

   `~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-deployment/zST/models/rf_model/1` directory

2. Add your preprocessing .joblib file to

   `~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-deployment/zST/models/rf_model/1` directory



Access sample code

Integrate AI model into Triton Inference Server

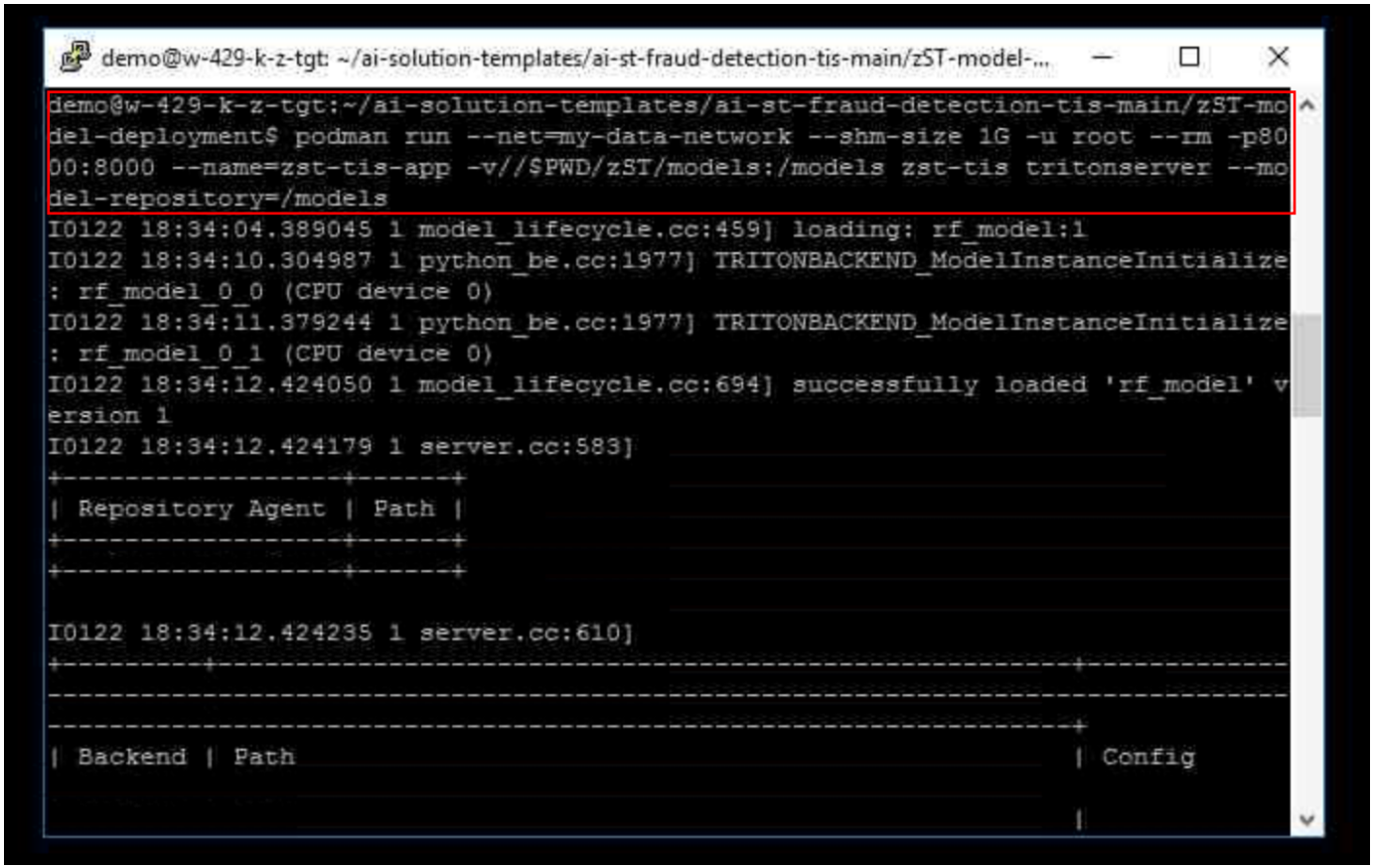
Deploy Triton Inference Server

Run sample test

Deploy Triton Inference Server

1. Run podman container

```
podman run --net=my-data-network --shm-size 1G -u root --rm -p8000:8000 --name=zst-tis-app -v//$PWD/zST/models:/models zst-tis tritonserver --model-repository=/models
```

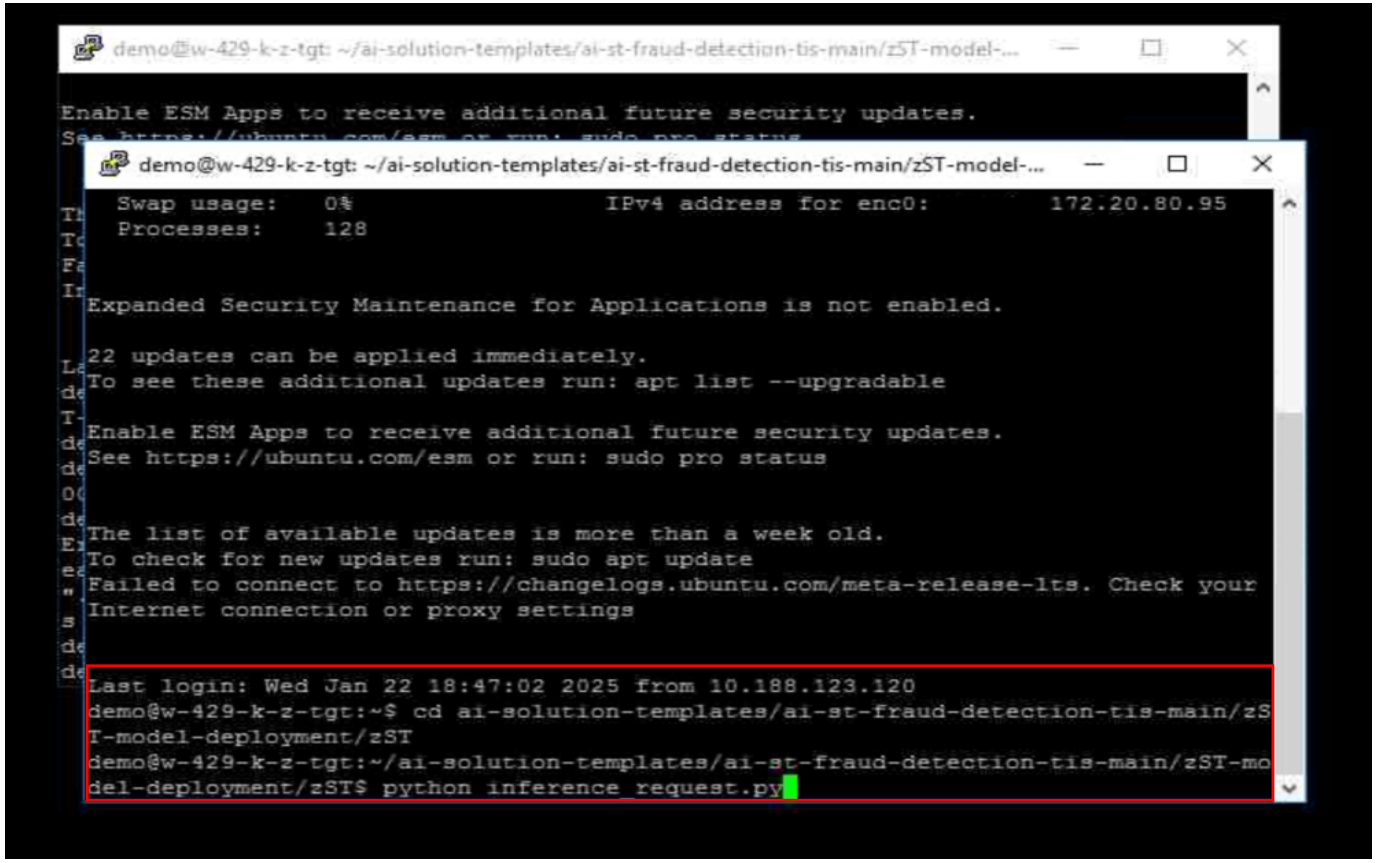


Run sample test

1. Run python script from terminal with ip/port of triton inference server (in new terminal)

```
cd ~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-deployment/zST
```

```
python inference_request.py
```



1. AI model training. 2. AI model deployment 3. AI model analysis 4. AI model integration

☑ AI model deployment complete

Environments used

Linux on Z
Windows

Step 3

AI model analysis

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

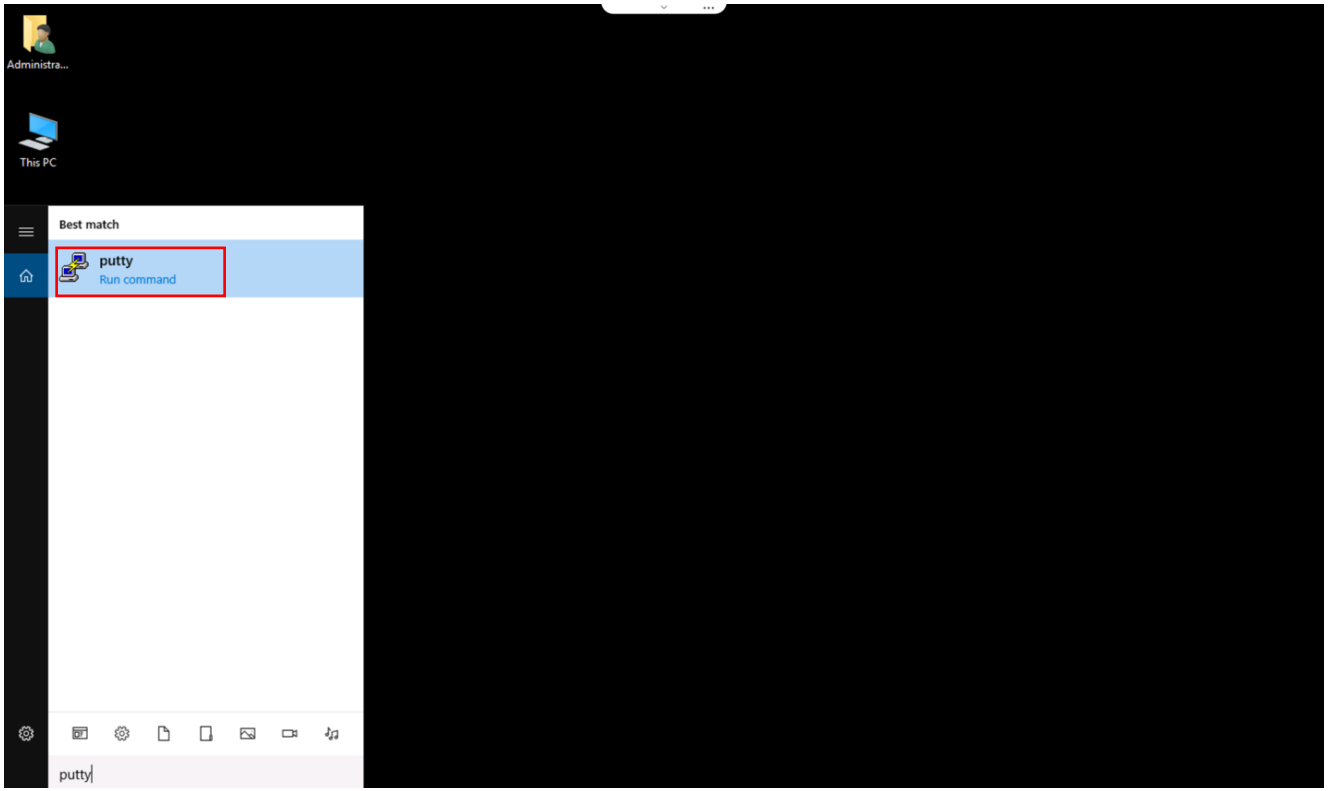
All sample code for this section is within

ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-analysis

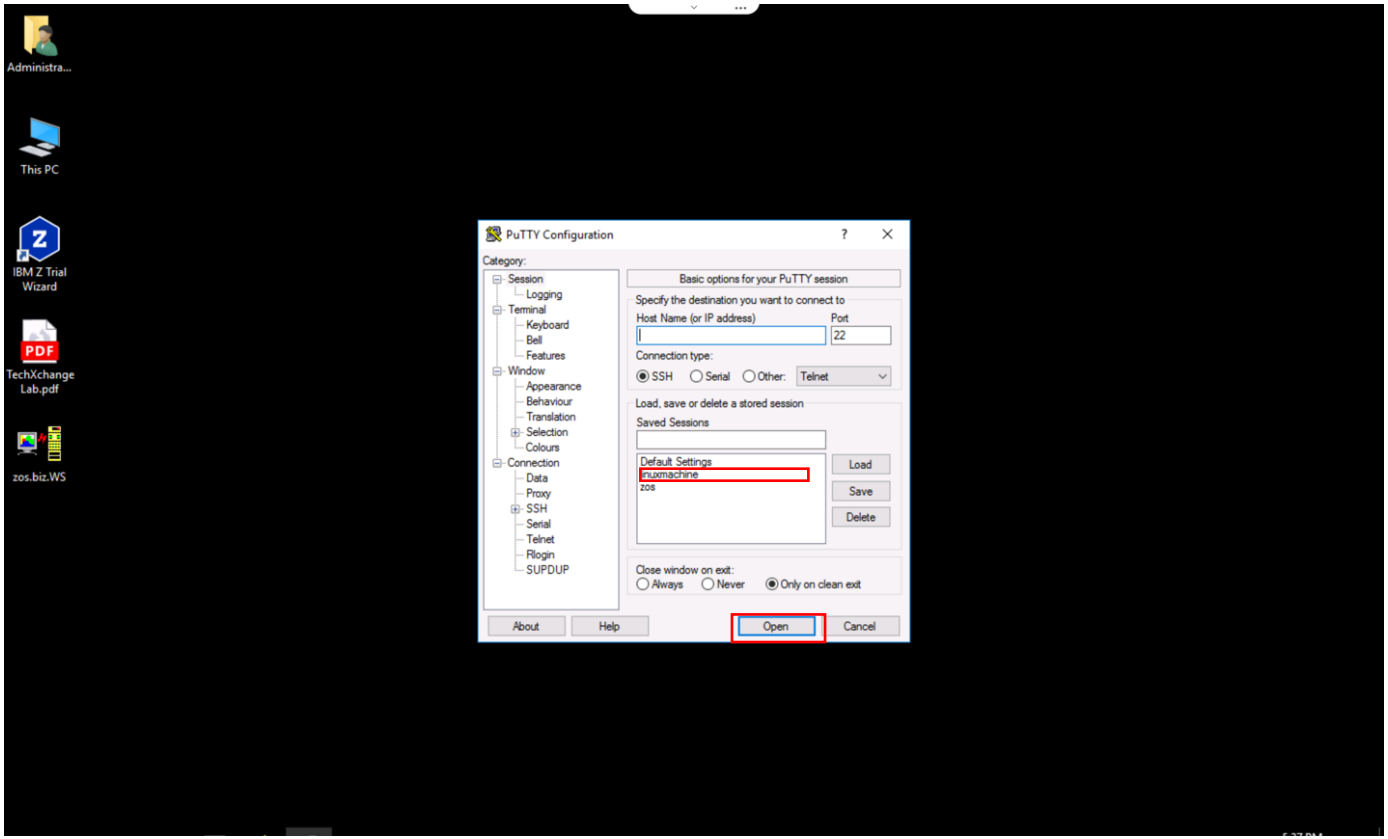
Access sample code
Configure sample application
Build sample application
Access sample application
Analyze credit card events
Make predication

Access sample code

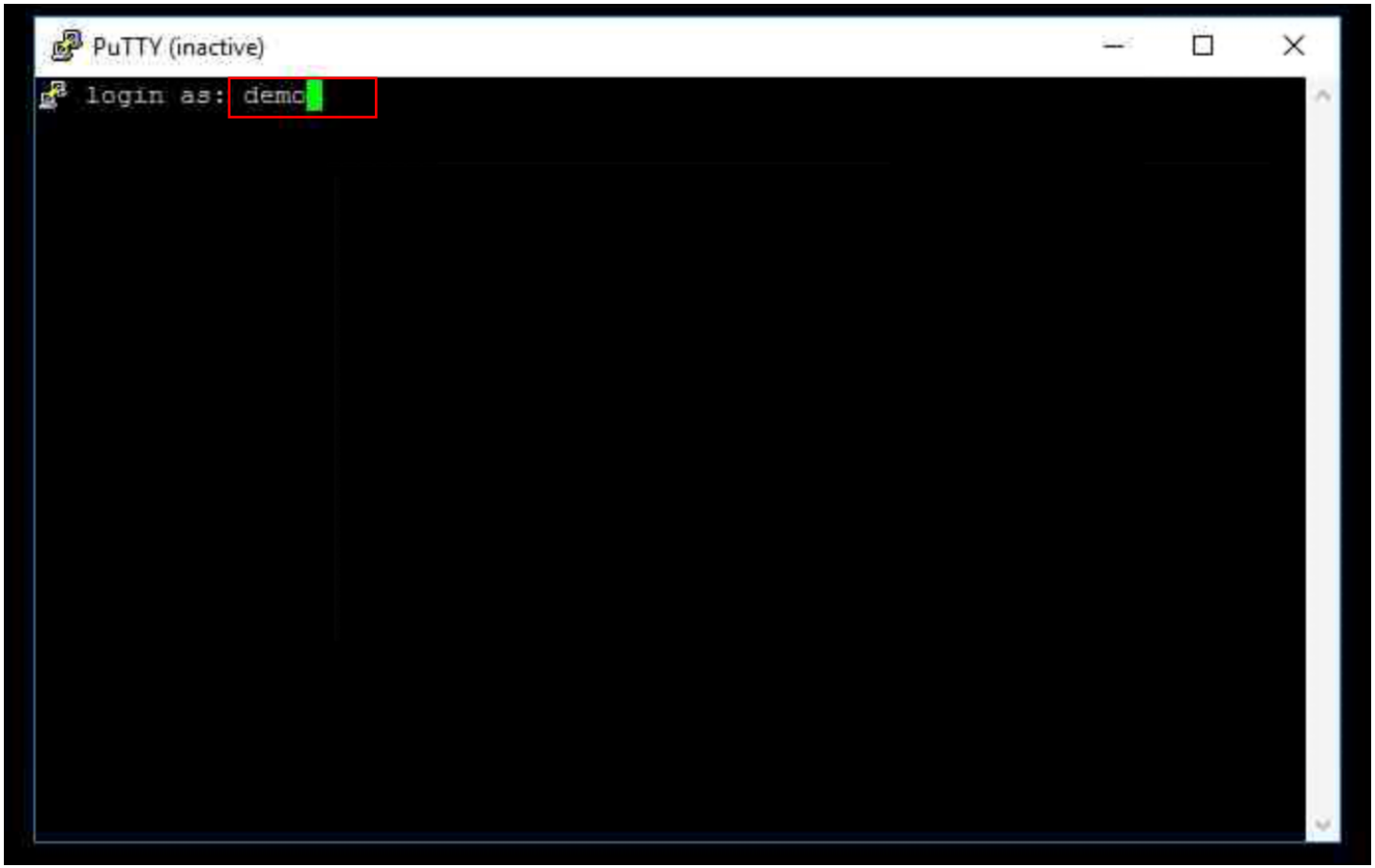
1. Open PuTTY application



2. Double click on linuxmachine

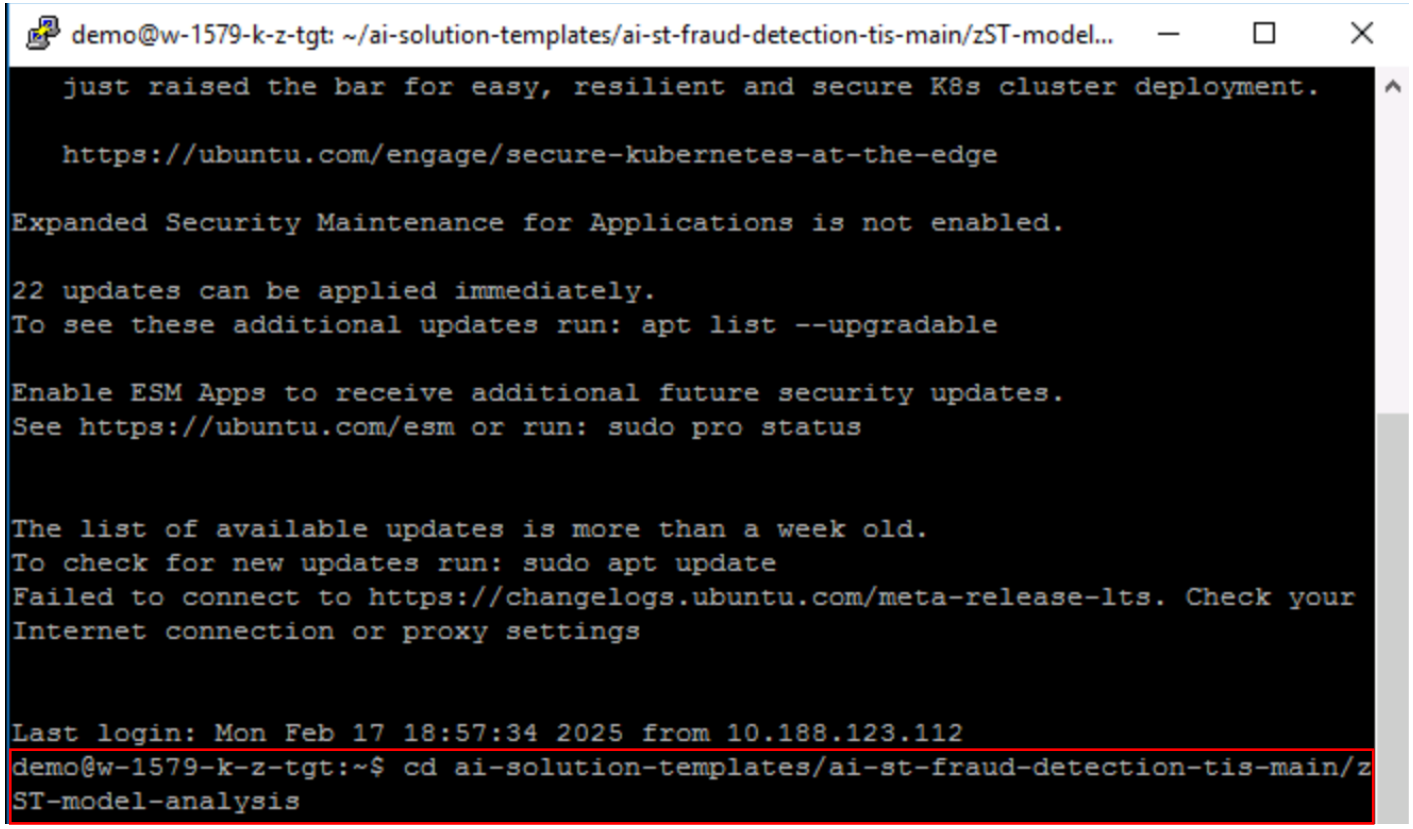


3. Enter demo username



4. Move to sample code directory

   ```
   cd ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-analysis
   ```

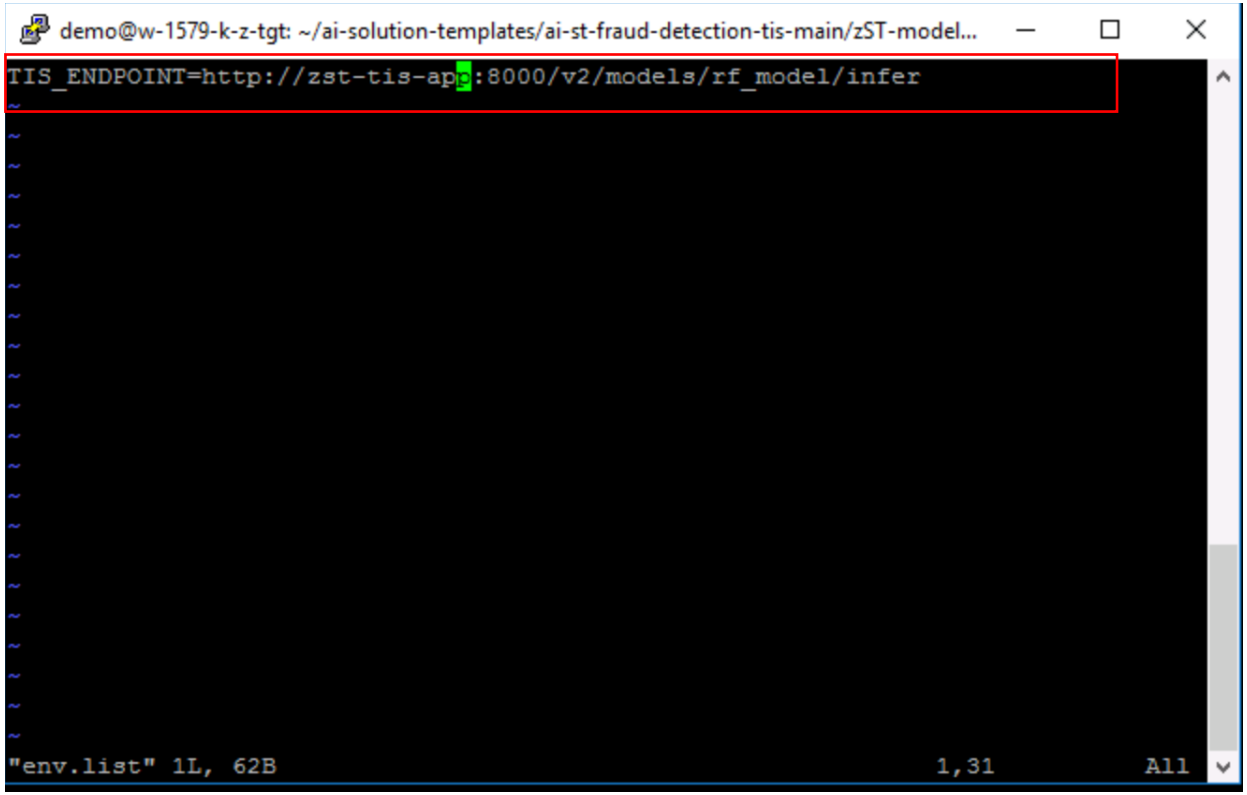


Configure sample application

1. Set the environment variables within

   `~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-analysis/env.list`

   TIS_ENDPOINT (scoring endpoint for deployed AI model)

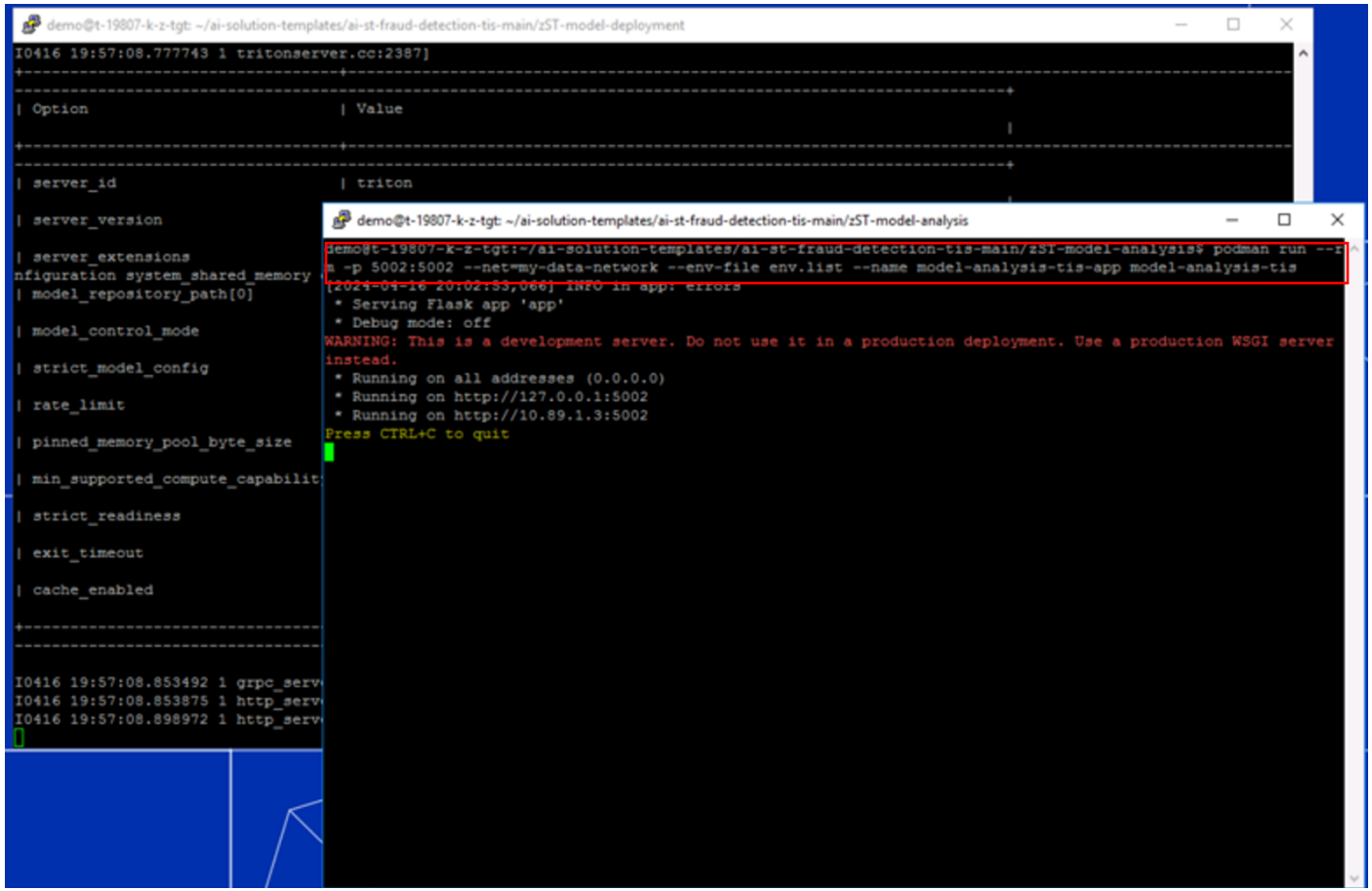


Get model details for inferencing
Access sample code
Configure sample application
Deploy sample application
Access sample application
Analyze credit card events
Make predication

Deploy sample application

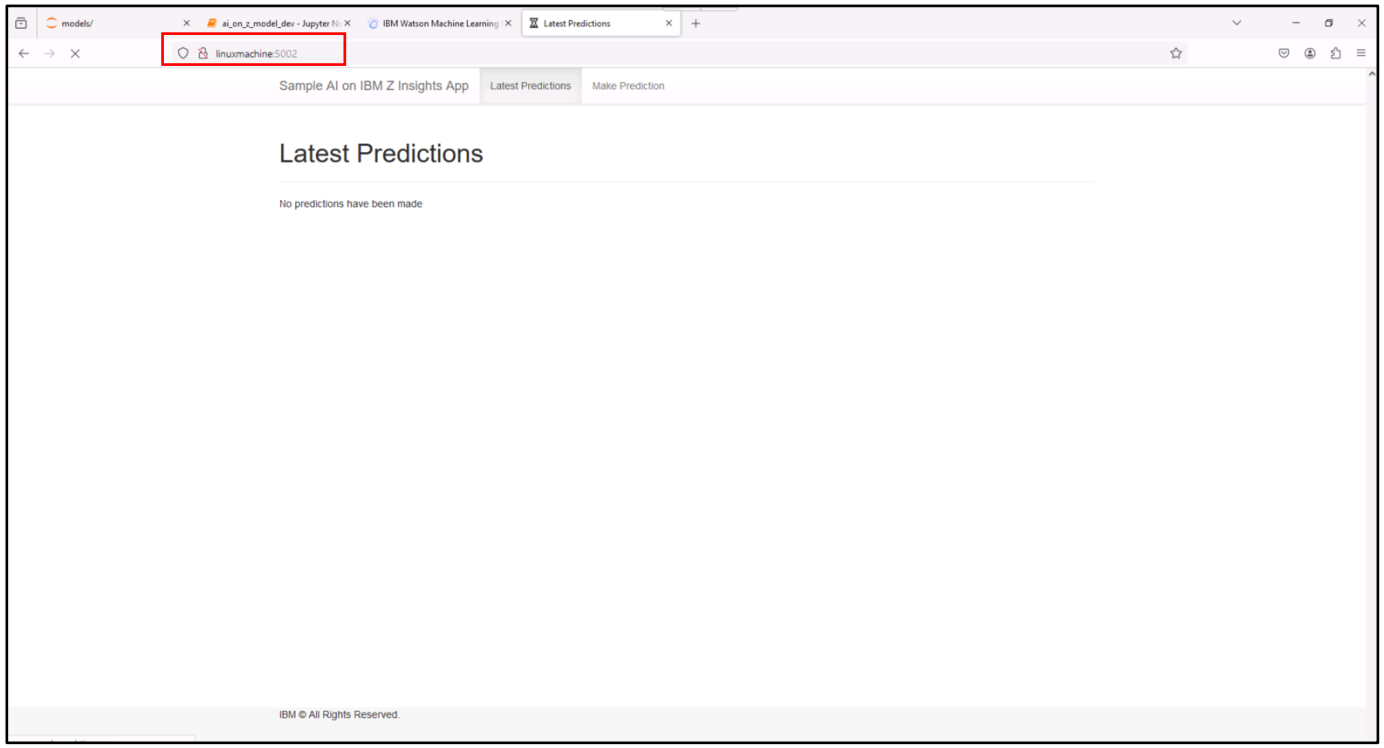
1. Run command in terminal (e.g. port 5002)

```
podman run --rm -p 5002:5002 --net=my-data-network  --env-file env.list --name model-analysis-tis-app model-analysis-tis
```



Access sample application

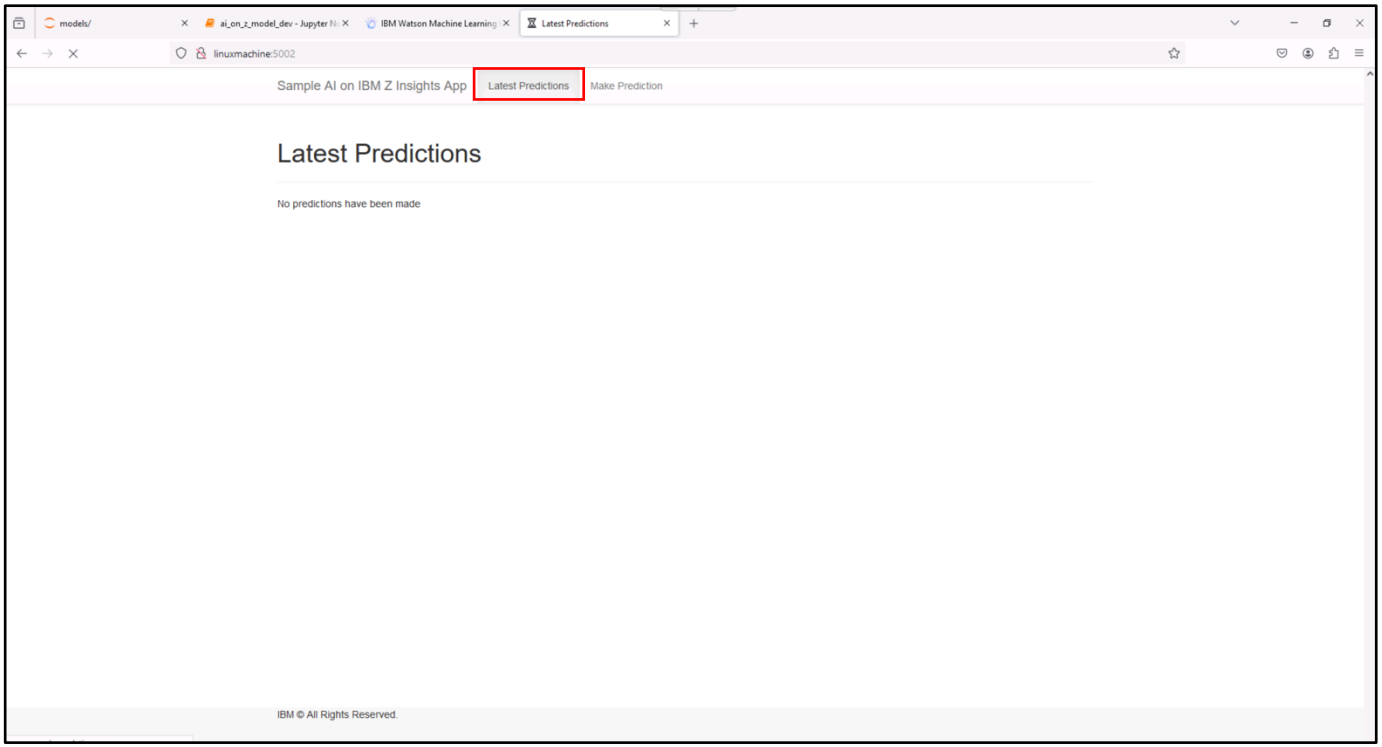
1. View the following URL in a web browser
   http://linuxMachine:{port}
   a. Port: port you used with podman run



Get model details for inferencing
Access sample code
Configure sample application
Deploy sample application
Access sample application
Analyze credit card events
Make predication

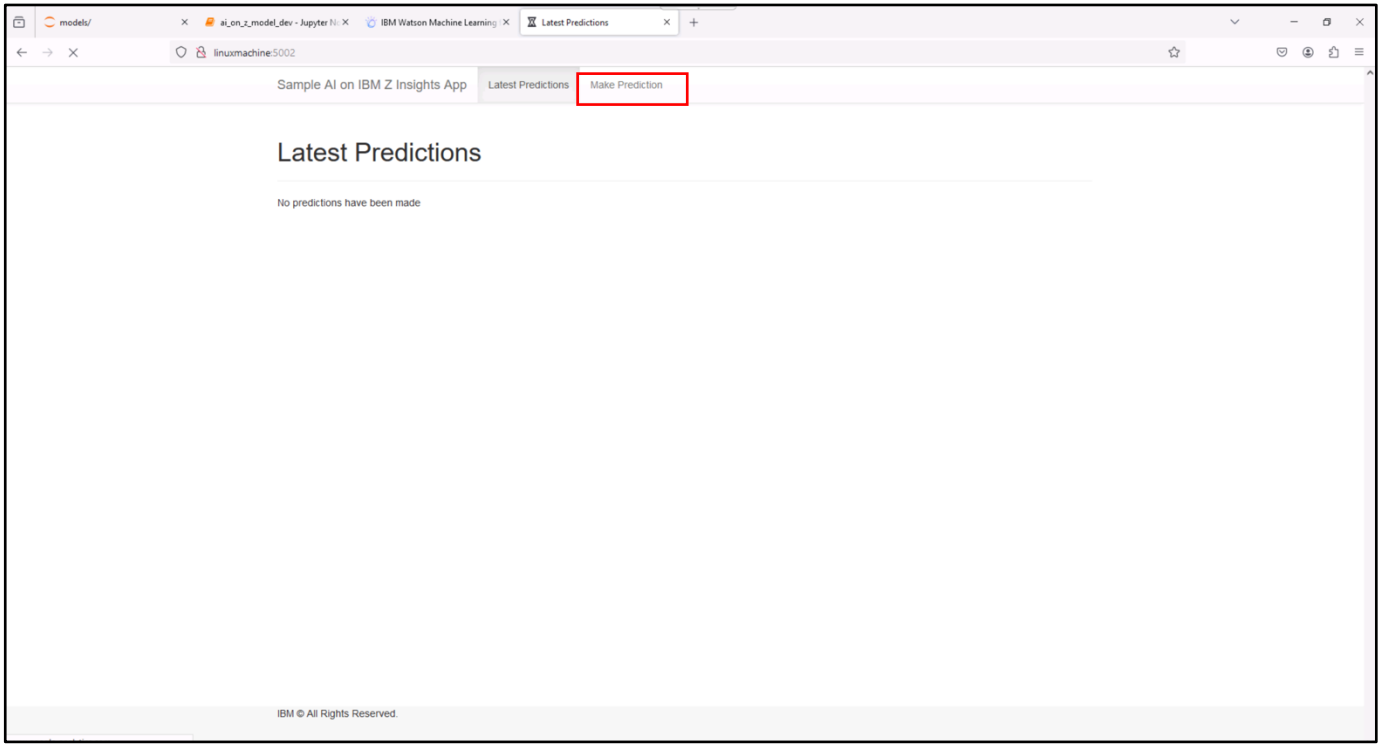
Analyze credit card events

1. Go to latest predictions tab

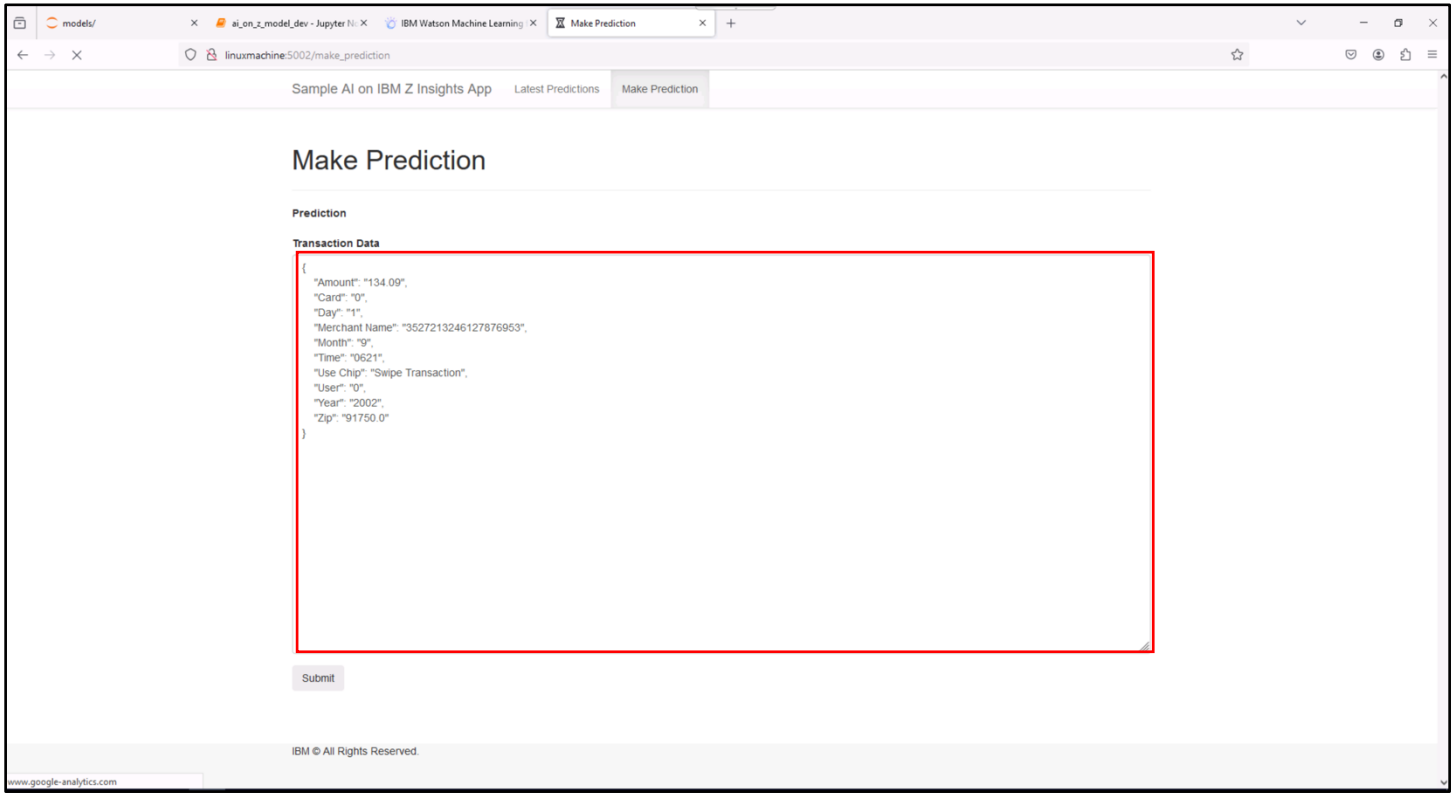


Make predication

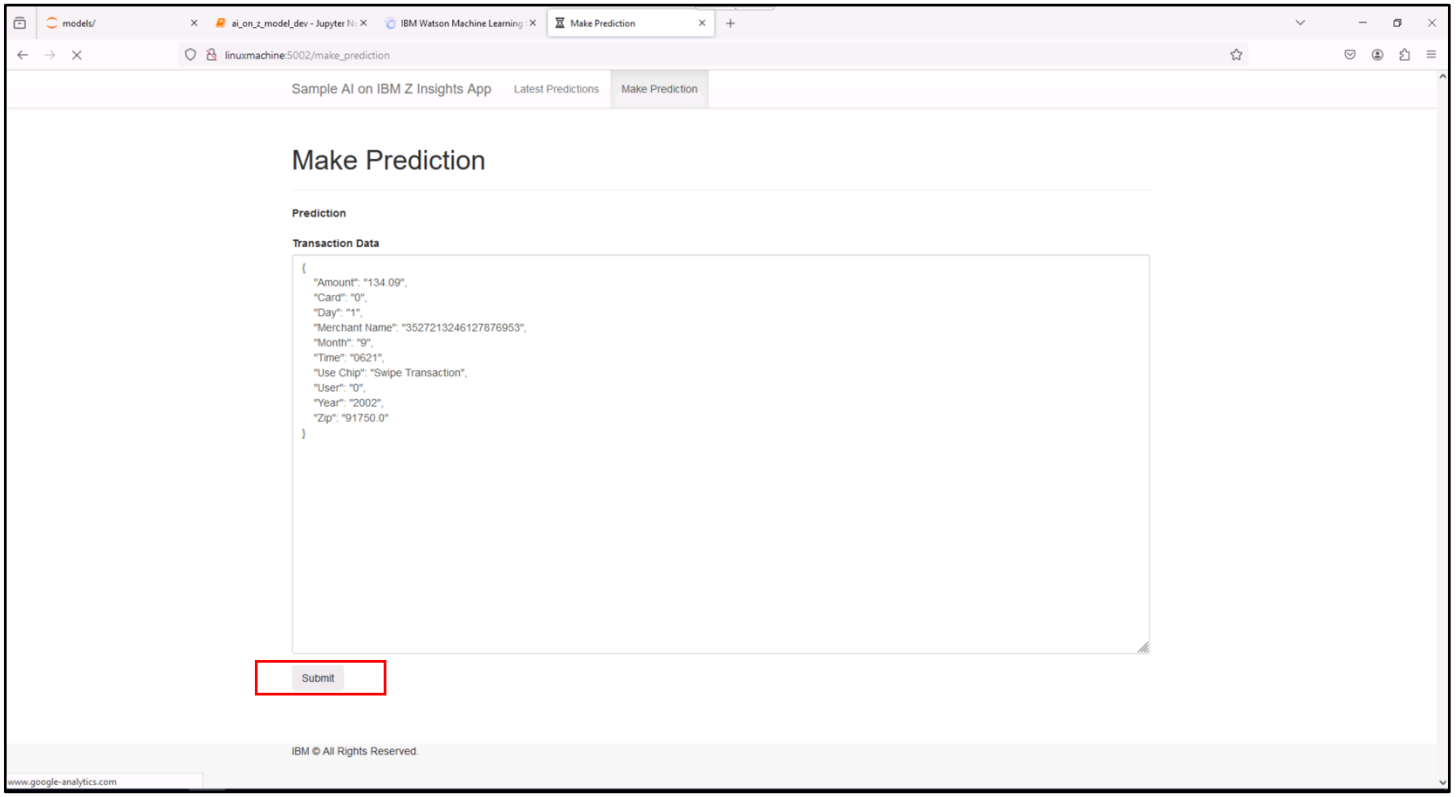
1. Go to latest make predication tab



2. Input json data in following format:
    a. {"Amount": "134.09", "Card": "0","Day": "1", "Merchant Name": "3527213246127876953", "Month": "9", "Time": "0621","Use Chip": "Swipe Transaction", "User": "0", "Year": "2002", "Zip": "91750.0"}



3. Click submit



1. AI model training. 2. AI model deployment 3. AI model analysis 4. AI model integration

☑ AI model analysis complete

Step 4

AI model integration

[Temporarily Removed]

We can use our deployed TIS 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 Linux on Z Fraud Detection Dashboard.

All sample code for this section is within

`ai-solution-templates/ai-st-fraud-detection-tis-main/zST-storefront-evershop`

Environments used

Linux on Z
Windows

Prerequisites

- Must have AI on IBM Z Sample Fraud Detection Dashboard deployed for inferencing and analysis
- Must have Docker or Podman installed

Access sample code

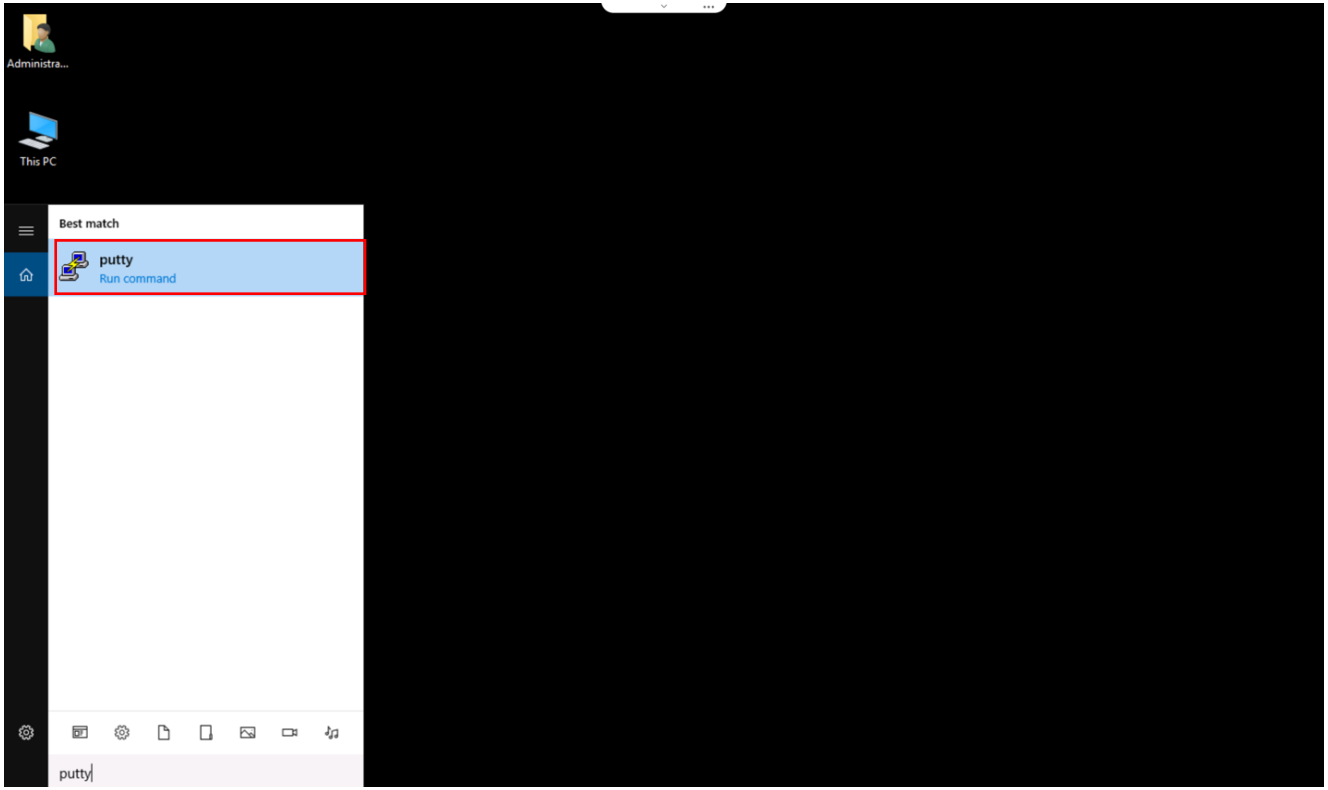
Deploy sample ecommerce application

Access sample ecommerce application

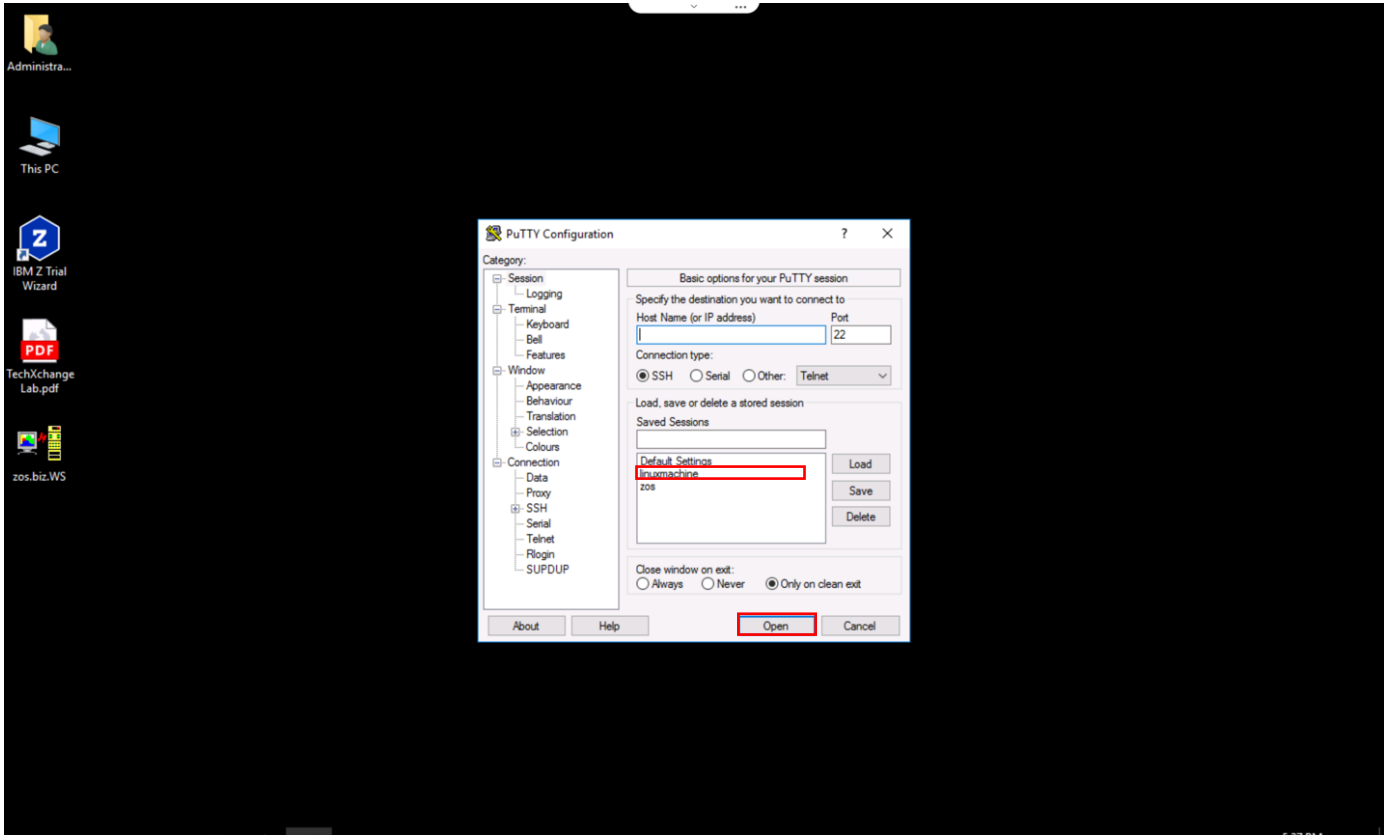
Use fraud detection AI model with EverShop Storefront

Access sample code

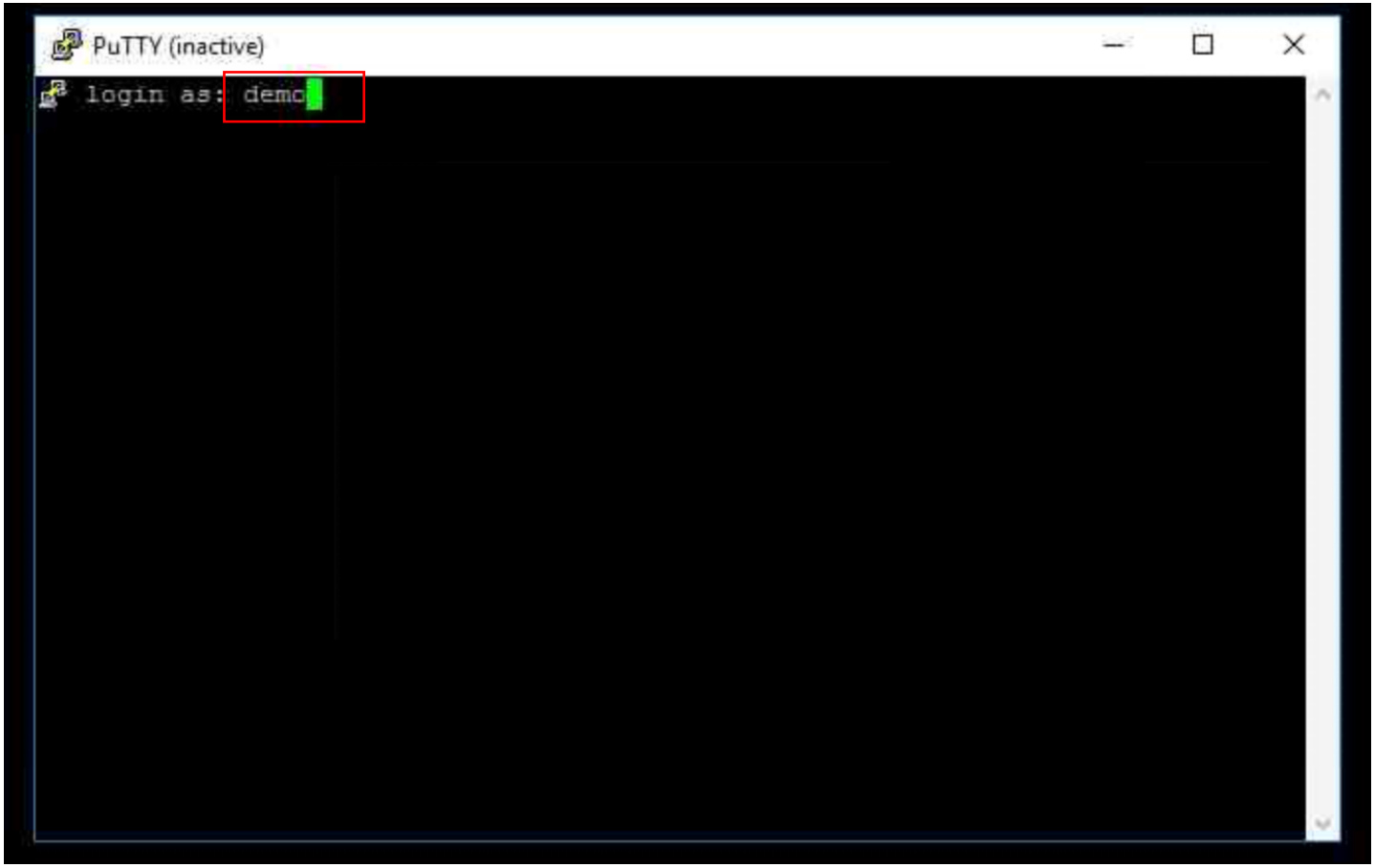
1. Open PuTTY application



2. Double click on linuxmachine

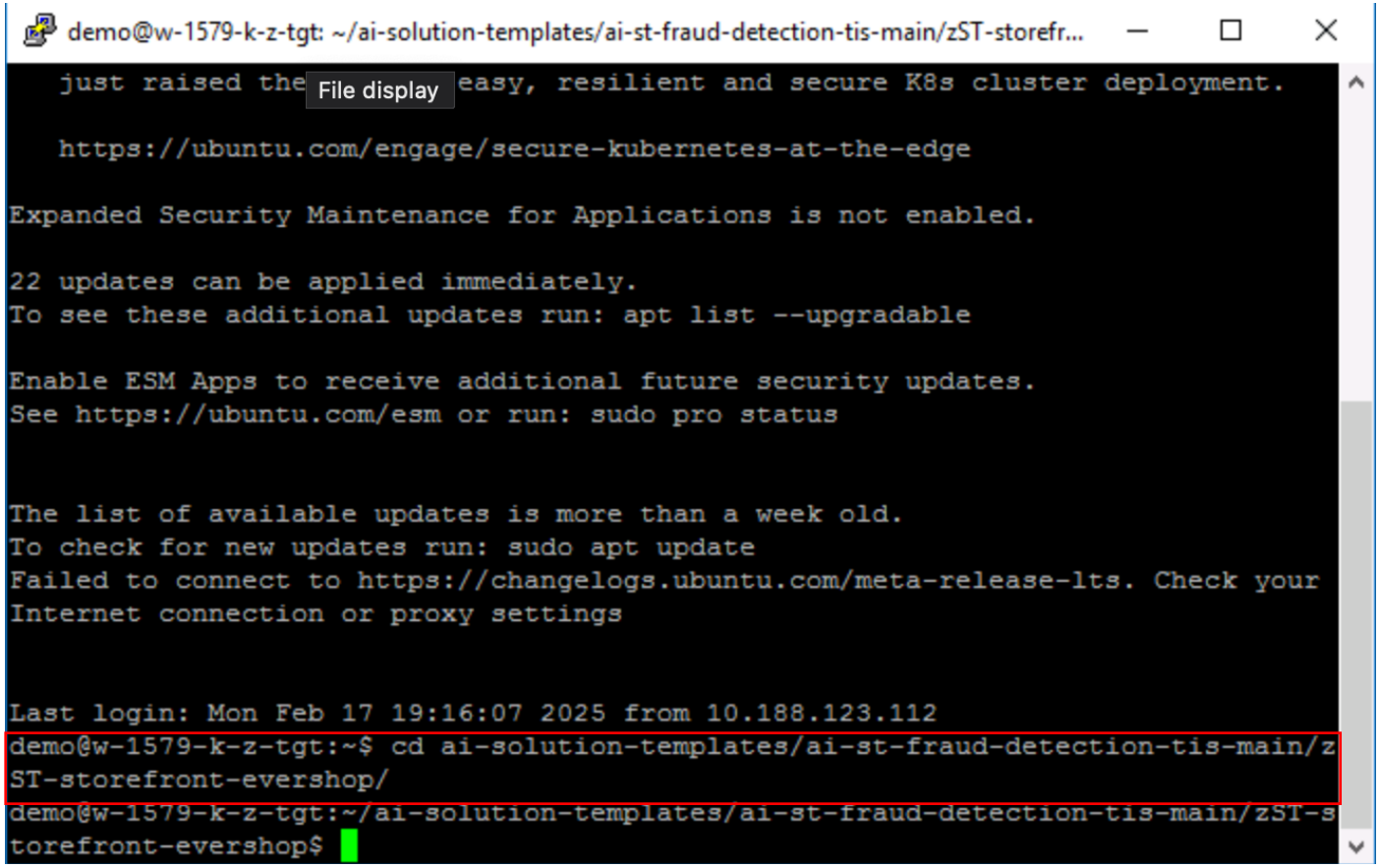


3. Enter demo username



4. Move to sample code directory

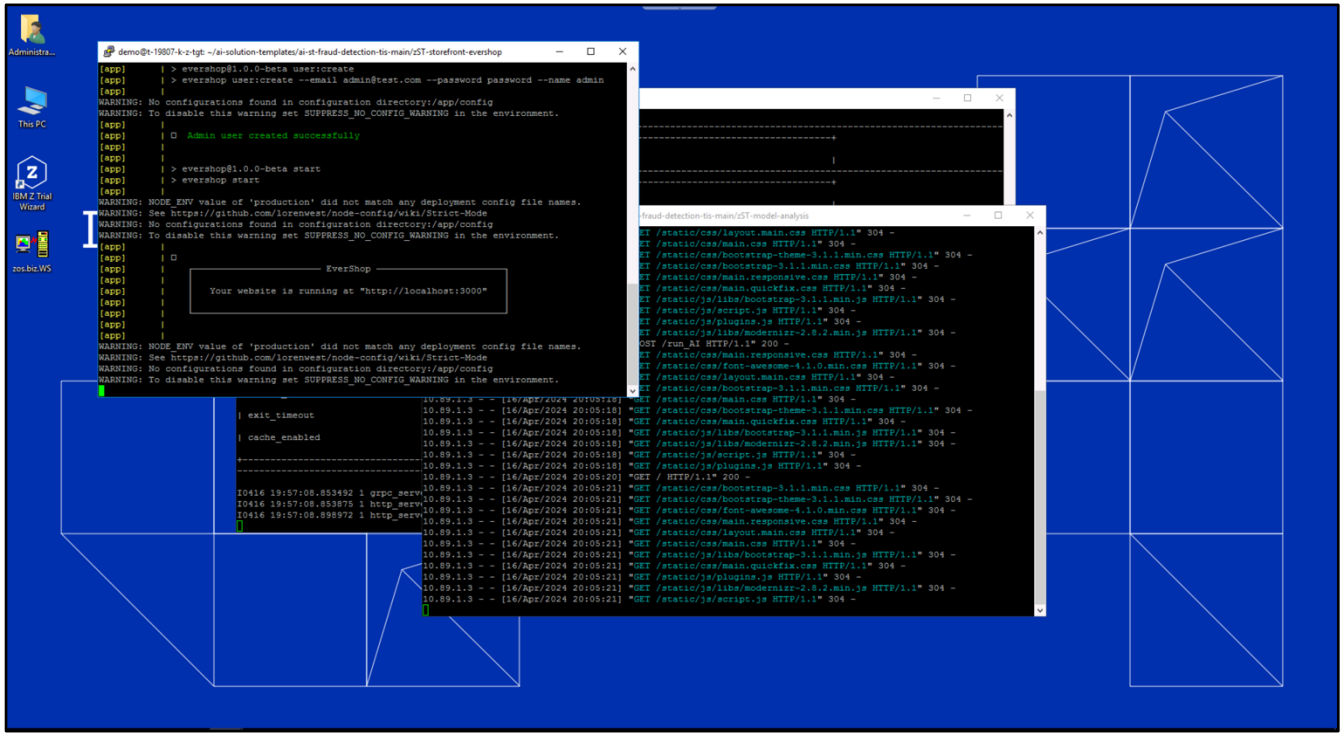
   cd ai-solution-templates/ai-st-fraud-detection-tis-main/zST-storefront-evershop



Deploy sample ecommerce application

1. Run command in terminal (password: demo)

   ```
   sudo podman-compose up --no-build
   ```



Access sample code

Deploy sample ecommerce application

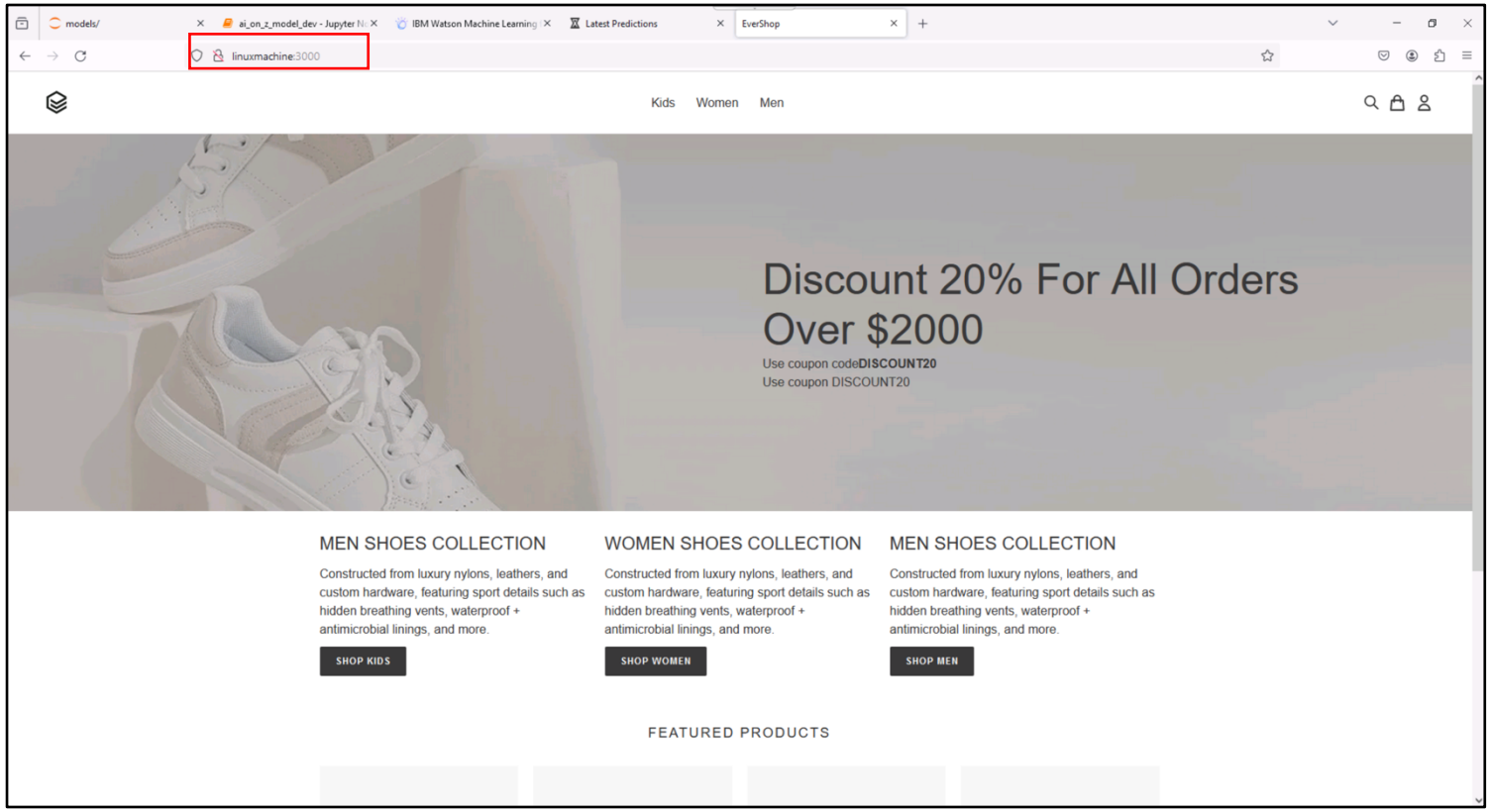
Access sample ecommerce application

Use fraud detection AI model with EverShop Storefront

Access sample ecommerce application

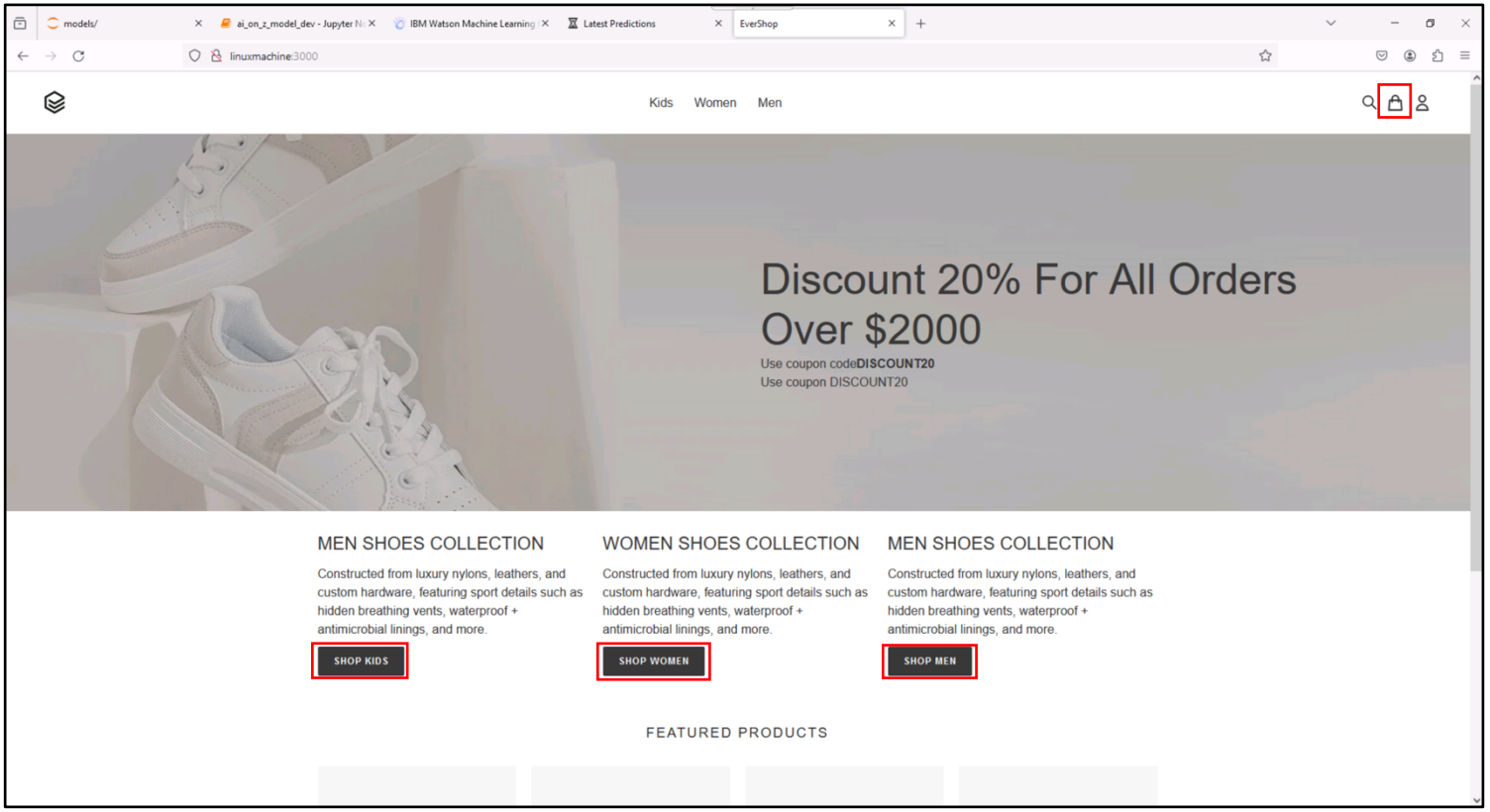
1. Enter URL in web browser using app URL

   http://linuxMachine:3000



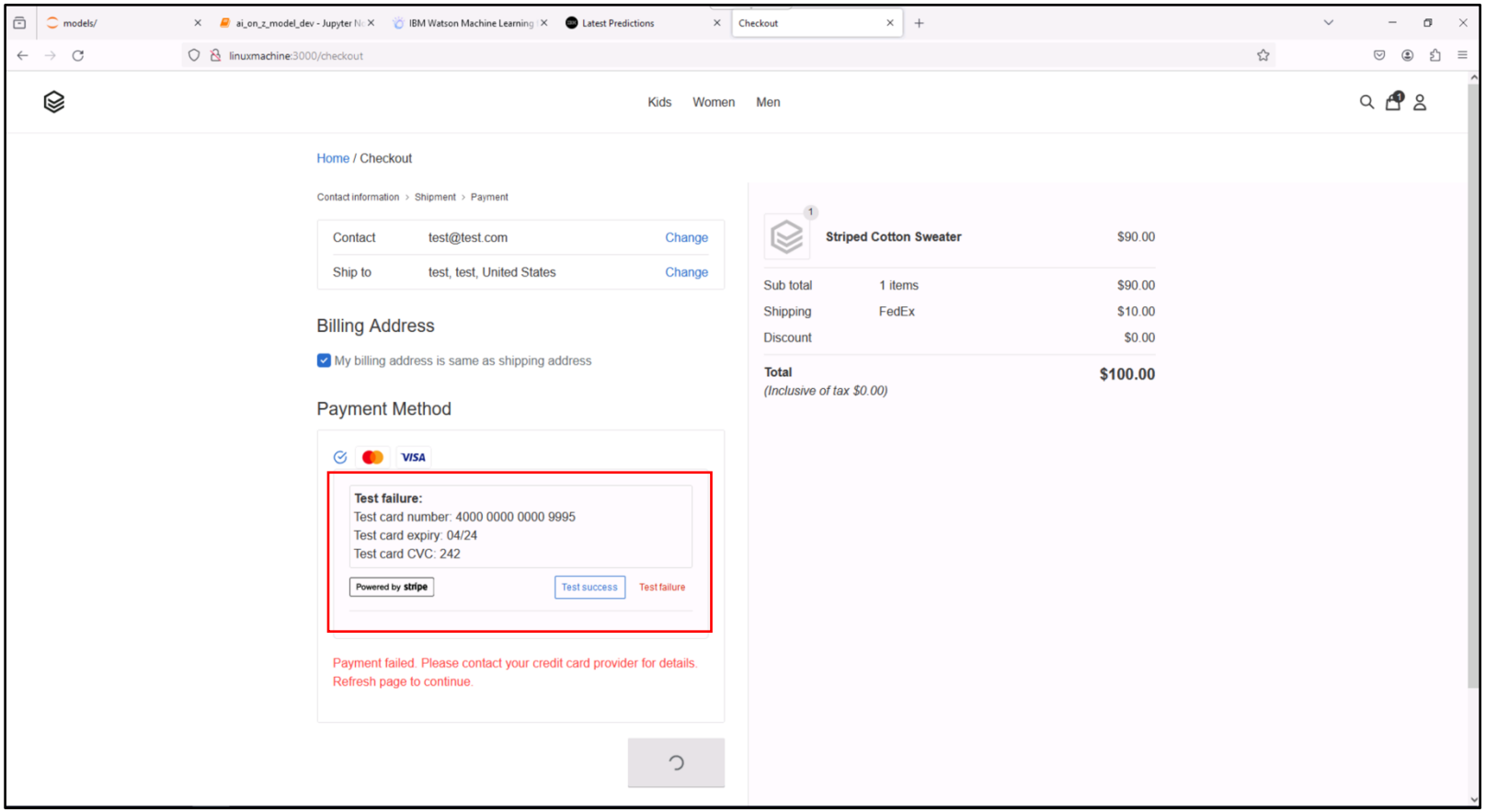
Use fraud detection AI model with EverShop Storefront

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

    a. 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