



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

1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

[Access rapid AI on IBM Z development environment](#)

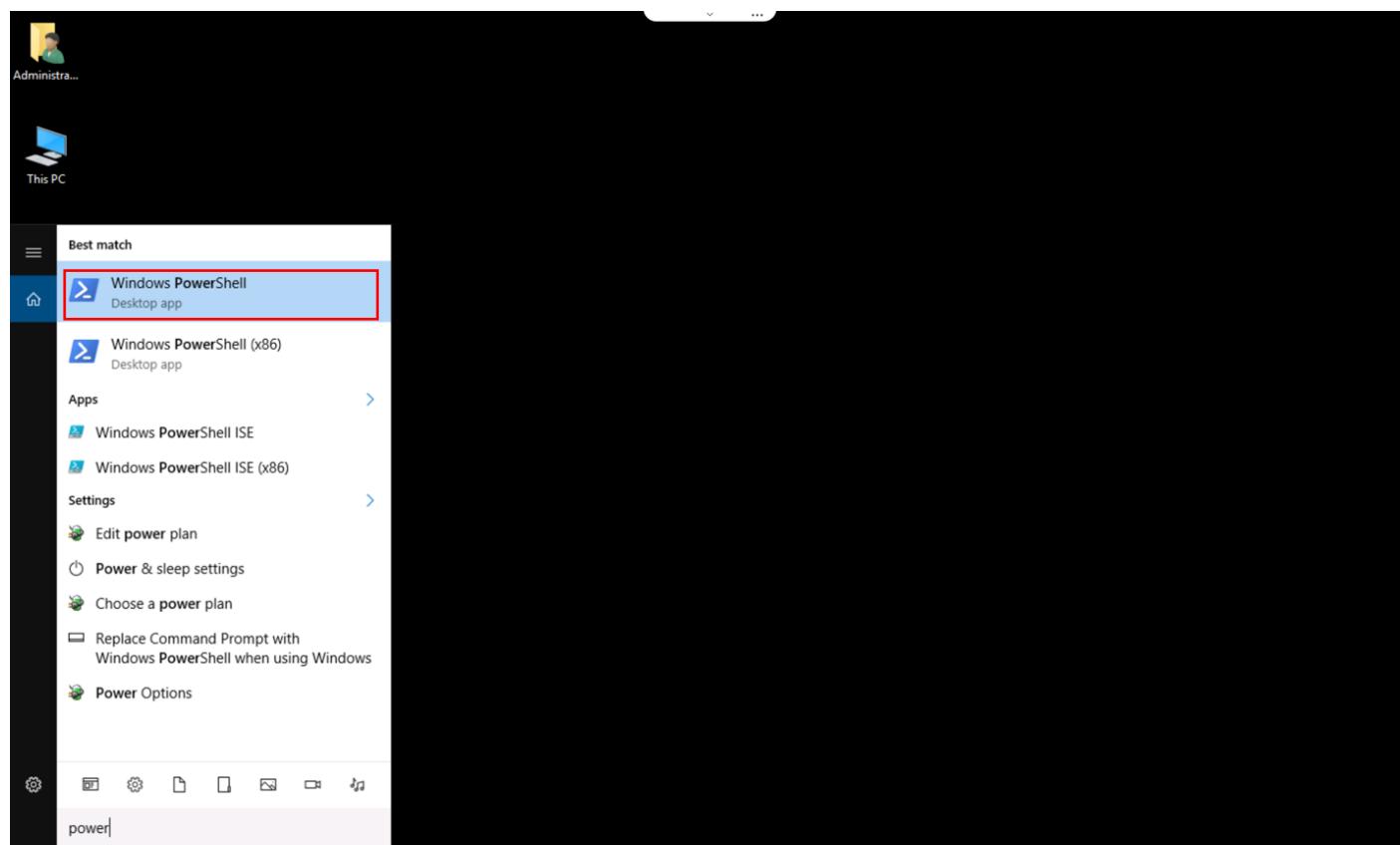
Provide data

Model training

Access trained AI model

Access rapid AI on IBM Z development environment

1. Open Windows PowerShell



1. AI model training.

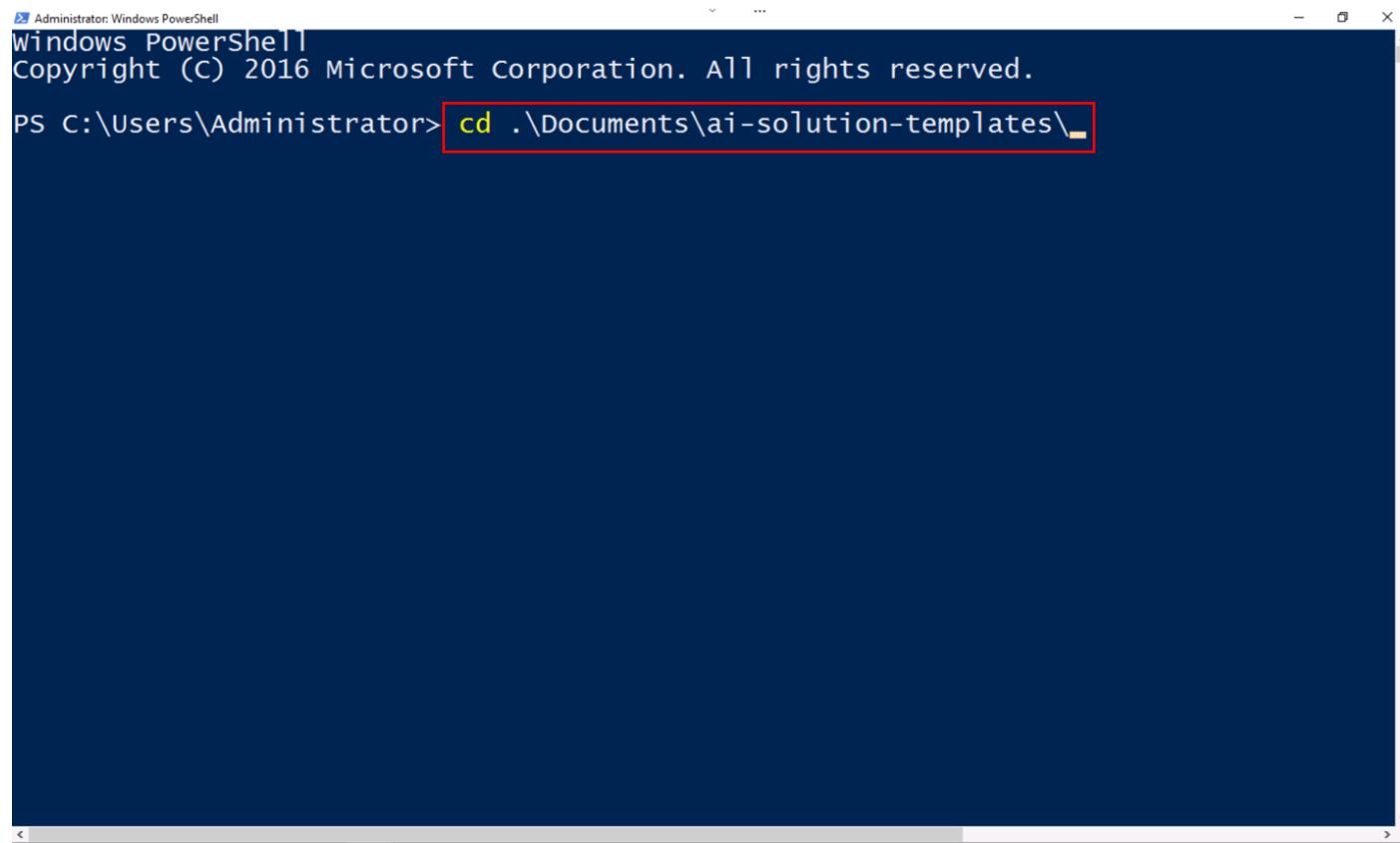
2. AI model deployment

3. AI model analysis

4. AI model integration

2. Go to model training pipeline

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



A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The window shows the command "cd .\Documents\ai-solution-templates\" entered at the prompt. The entire command line is highlighted with a red box.

1. AI model training.

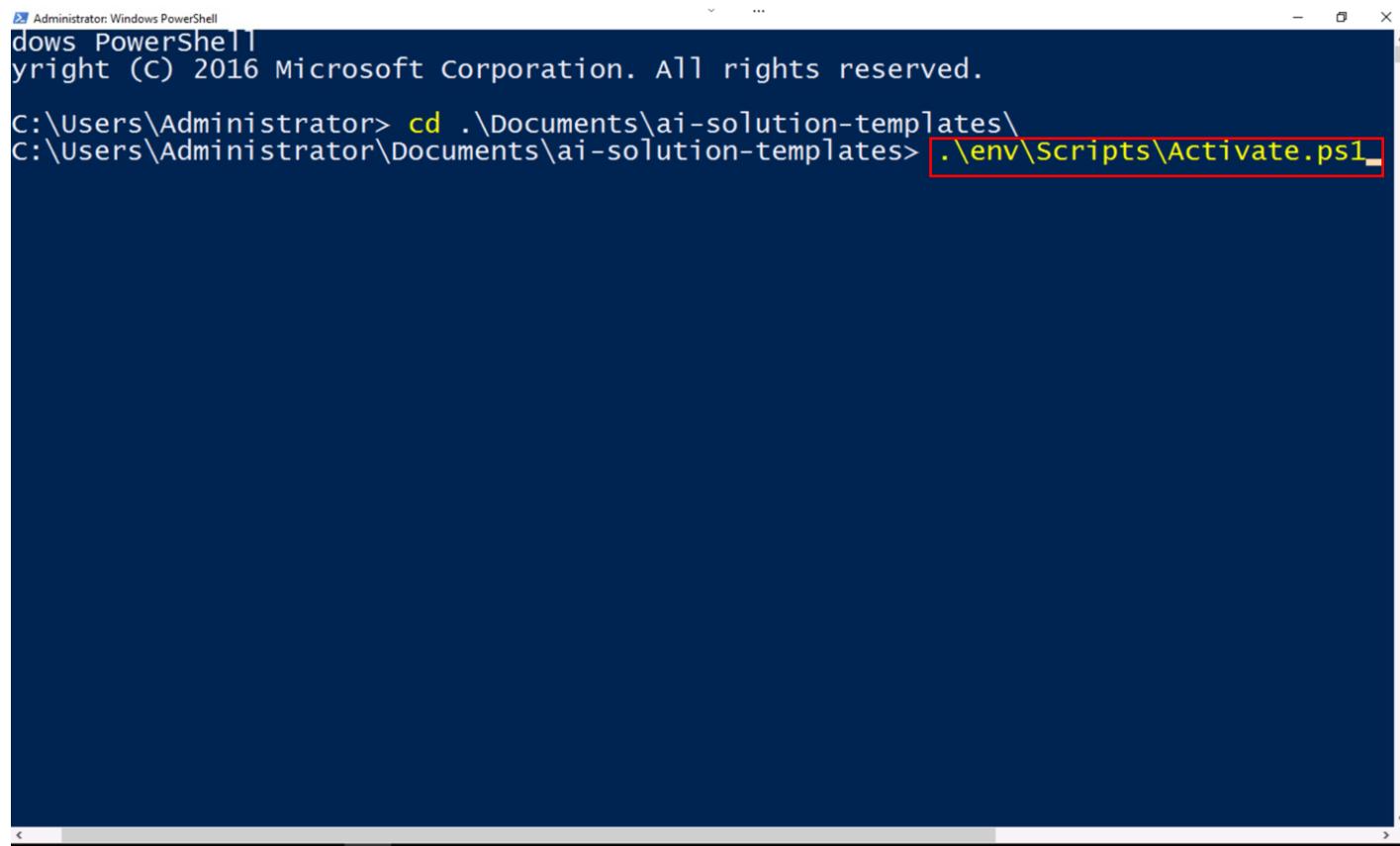
2. AI model deployment

3. AI model analysis

4. AI model integration

3. Activate Python virtual environment

`.\env\Scripts\Activate.ps1`



A screenshot of a Windows PowerShell window titled "Administrator Windows PowerShell". The window shows the following command being run:

```
Administrator Windows PowerShell
Windows PowerShell
Copyright (c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Administrator> cd .\Documents\ai-solution-templates\
C:\Users\Administrator\Documents\ai-solution-templates> .\env\Scripts\Activate.ps1
```

The command `.\env\Scripts\Activate.ps1` is highlighted with a red rectangle.

1. AI model training.

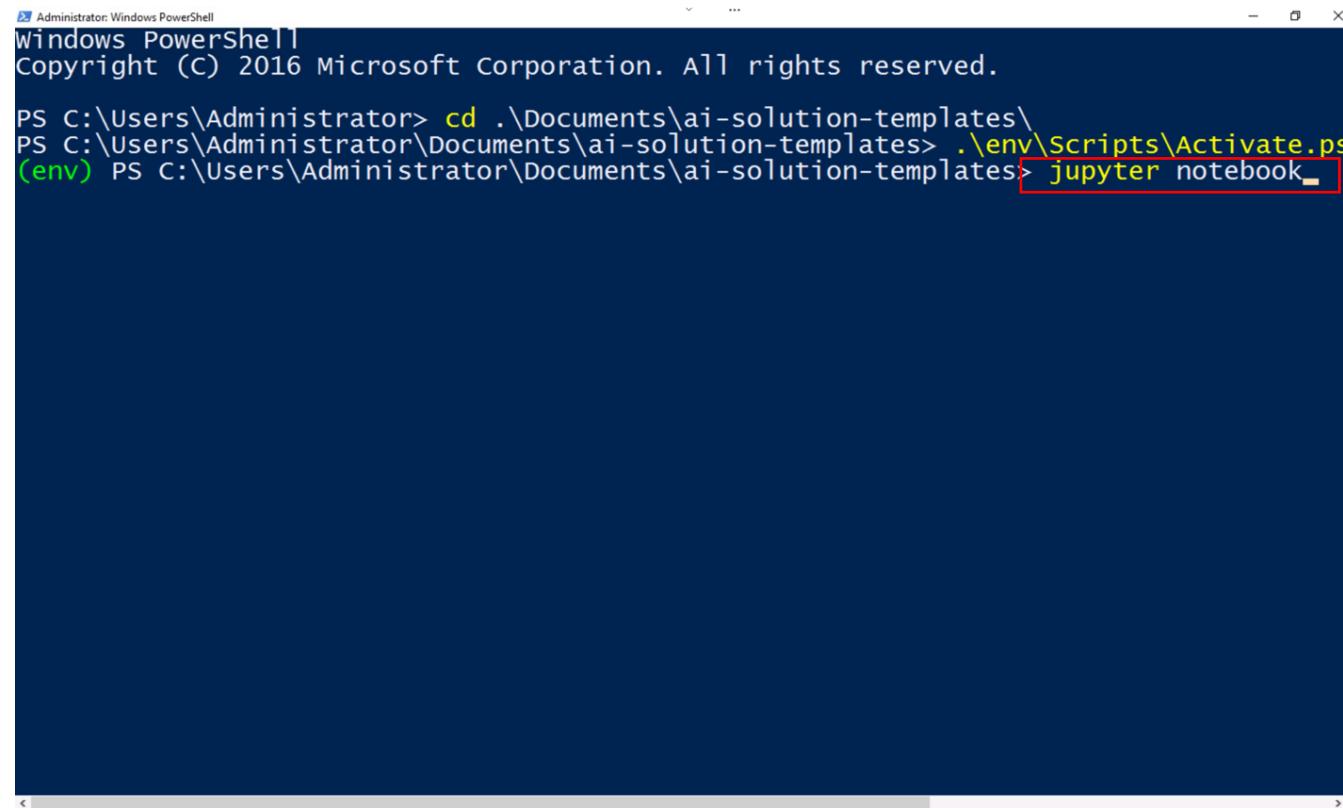
2. AI model deployment

3. AI model analysis

4. AI model integration

4. Run Jupyter

jupyter notebook



A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The window shows the following command sequence:

```
PS C:\Users\Administrator> cd .\Documents\ai-solution-templates\  
PS C:\Users\Administrator\Documents\ai-solution-templates> .\env\Scripts\Activate.ps1  
(env) PS C:\Users\Administrator\Documents\ai-solution-templates> jupyter notebook
```

The last command, "jupyter notebook", is highlighted with a red rectangular box.

1. AI model training.

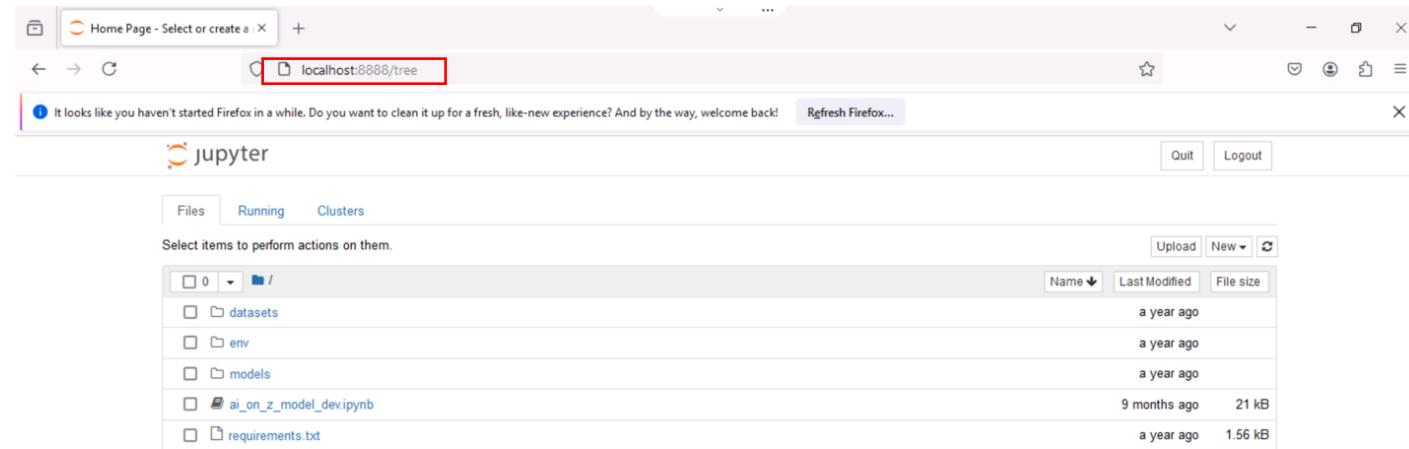
2. AI model deployment

3. AI model analysis

4. AI model integration

5. View Jupyter interface

- a. Go to localhost:8888 in a web browser



1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

Access rapid AI on IBM Z development environment

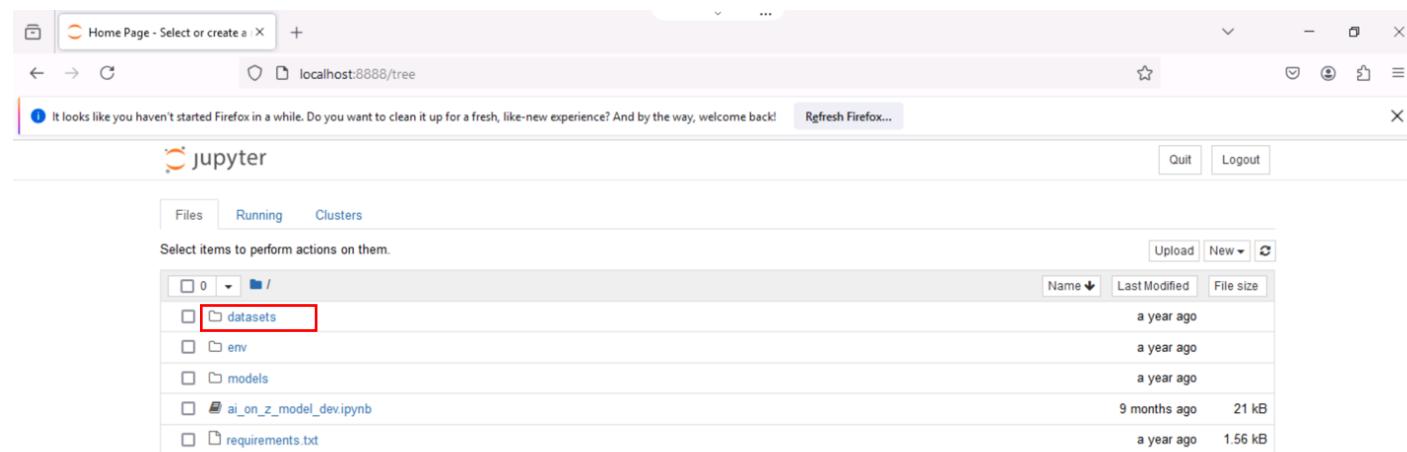
[Provide data](#)

[Model training](#)

Access trained AI model

Provide data (optional)

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



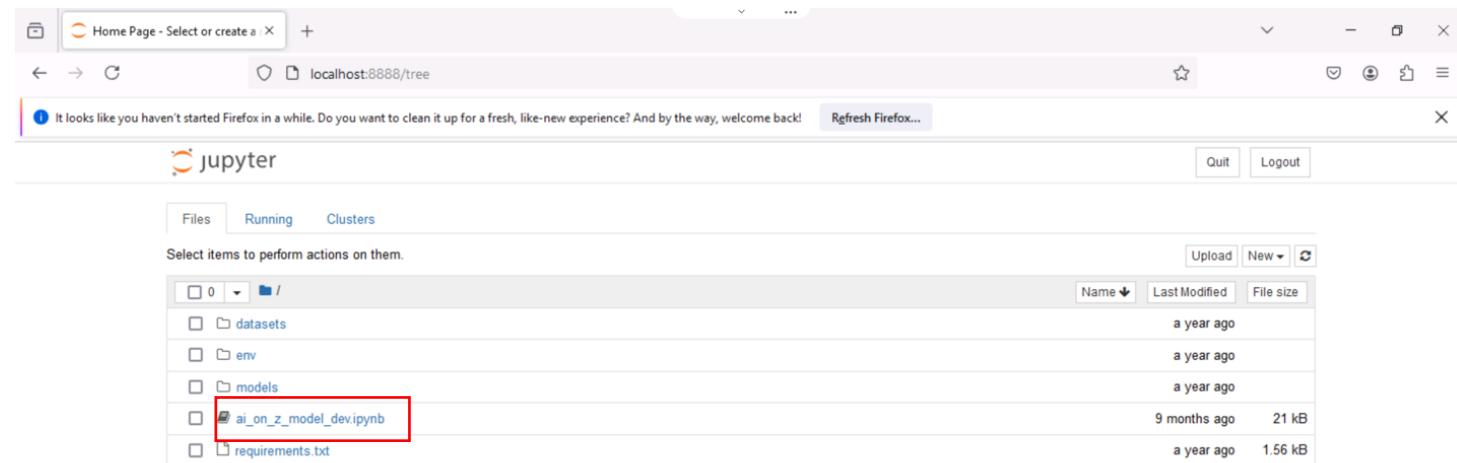
1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

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

The screenshot shows a Jupyter Notebook interface with the title "AI on IBM Z Model Development". The notebook contains three sections of code:

- Import required python packages**:

```
In [ ]: import numpy as np
import pandas as pd
import json
import time

# Model training
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, GradientBoostingRegressor, RandomForestRegressor
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score

# Data preprocessing
from sklearn.preprocessing import OrdinalEncoder, StandardScaler

# PMML
from sklearn2pmml import sklearn2pmml
from sklearn.compose import ColumnTransformer
from sklearn2pmml.pipeline import PMMLPipeline
```
- Input dataset and label**:

```
In [ ]: # User must provide filepath to dataset and label name
# Available datasets / label names:
# - Credit_card_transactions-ibm_v2.csv / Is Fraud?
# - Data_Premium_V1.csv / Insurance_cost
# - credit_risk_dataset.csv / loan status
DATASET_FILENAME = 'datasets/credit_card_transactions-ibm_v2.csv'
DATASET_LABEL_NAME = 'Is Fraud?'
```

The line `DATASET_FILENAME = 'datasets/credit_card_transactions-ibm_v2.csv'` is highlighted with a red box.
- Split features and labels from dataset**:

```
In [ ]: def split_features_and_labels(dataset_df, label):
    features = dataset_df.copy()
```

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 screenshot shows a Jupyter Notebook interface titled "jupyter ai_on_z_model_dev". The notebook has a single cell containing Python code for AI model training. A context menu is open over the cell, with the "Run All" option highlighted by a red box. The code in the cell includes imports for numpy, pandas, json, time, Pipeline, GridSearchCV, GradientBoostingClassifier, RandomForestClassifier, GradientBoostingRegressor, RandomForestRegressor, precision_score, recall_score, f1_score, and accuracy_score. It also imports OrdinalEncoder and StandardScaler from sklearn.preprocessing, and PMMLPipeline from sklearn2pmml.pipeline. The code is divided into sections for Model training, Data preprocessing, and PMML.

```
import numpy
import pandas
import json
import time

# Model training
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier, GradientBoostingRegressor, RandomForestRegressor
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score

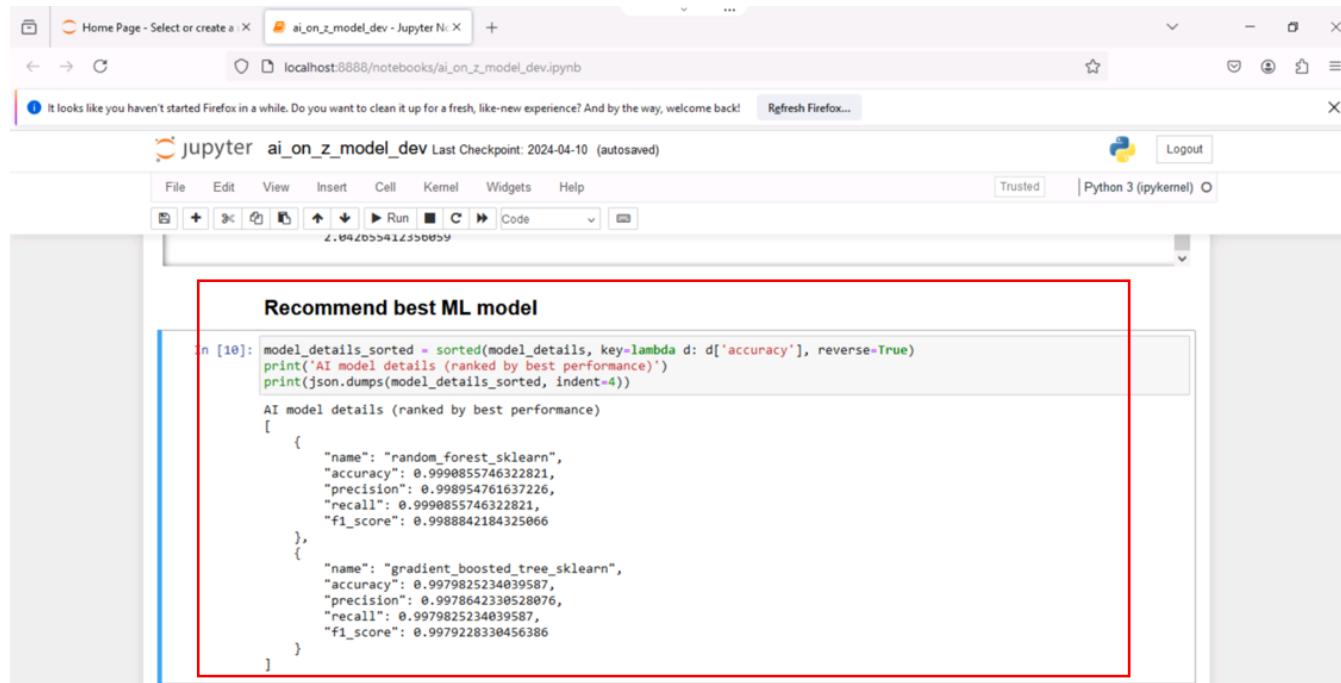
# Data preprocessing
from sklearn.preprocessing import OrdinalEncoder, StandardScaler

# PMML
from sklearn2pmml import sklearn2pmml
from sklearn.compose import ColumnTransformer
from sklearn2pmml.pipeline import PMMLPipeline
```

In []: # User must provide filepath to dataset and label name

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.



A screenshot of a Jupyter Notebook interface titled "jupyter ai_on_z_model_dev". The notebook shows a single cell with the following Python code:

```
In [10]: model_details_sorted = sorted(model_details, key=lambda d: d['accuracy'], reverse=True)
print('AI model details (ranked by best performance)')
print(json.dumps(model_details_sorted, indent=4))

AI model details (ranked by best performance)
[
    {
        "name": "random_forest_sklearn",
        "accuracy": 0.999855746322821,
        "precision": 0.998954761637226,
        "recall": 0.999855746322821,
        "f1_score": 0.9988842184325066
    },
    {
        "name": "gradient_boosted_tree_sklearn",
        "accuracy": 0.9979825234039587,
        "precision": 0.9978642330528076,
        "recall": 0.9979825234039587,
        "f1_score": 0.9979228330456386
    }
]
```

The output of the code is displayed in the cell below, showing the sorted list of AI models with their respective accuracy, precision, recall, and F1-score. A red box highlights the output area of the code cell.

Access rapid AI on IBM Z development environment

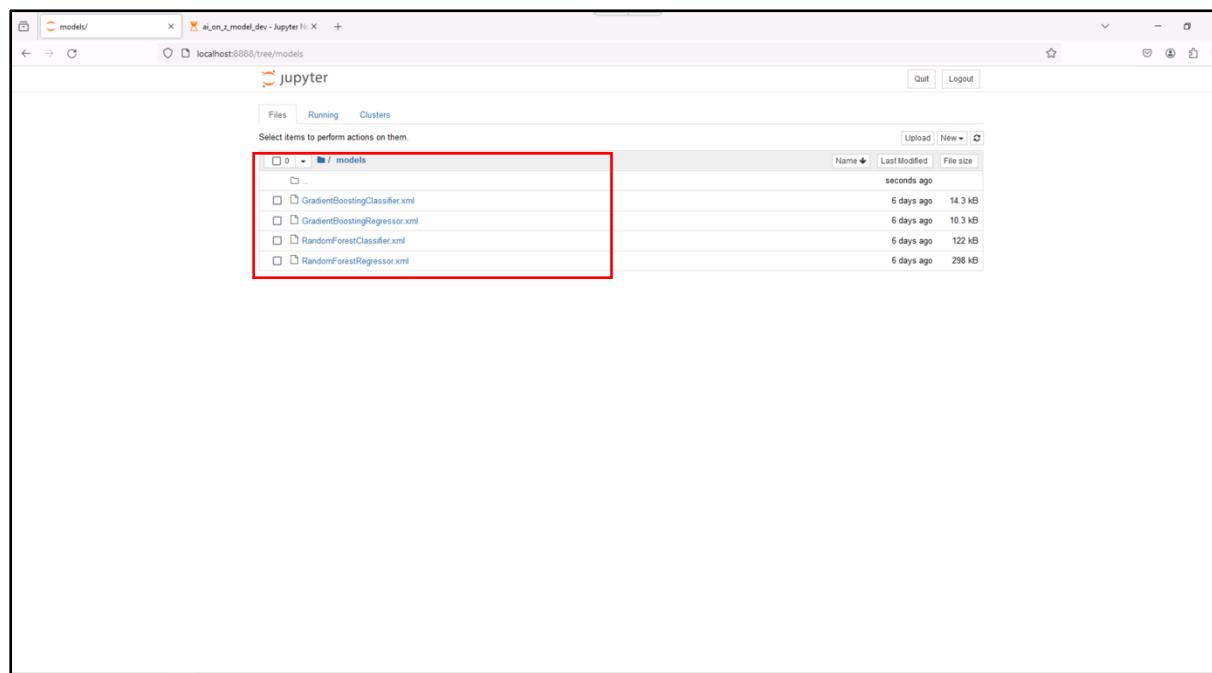
Provide data

Model training

[Access trained AI model](#)

Access trained AI model

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



1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

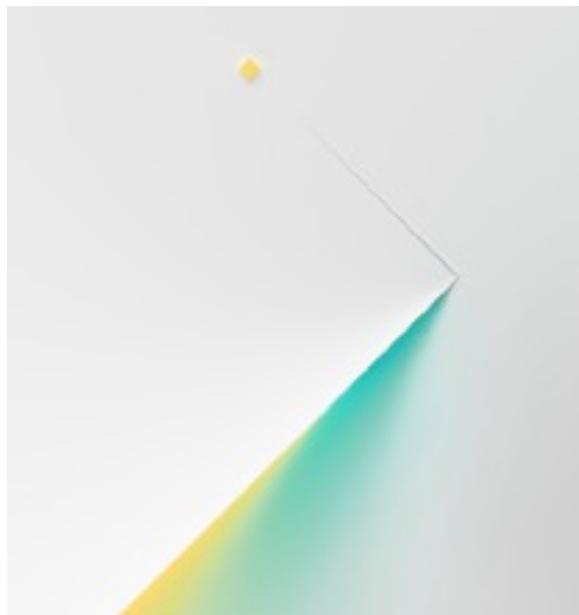
AI model training complete

1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration



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.

1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

[Access sample code](#)

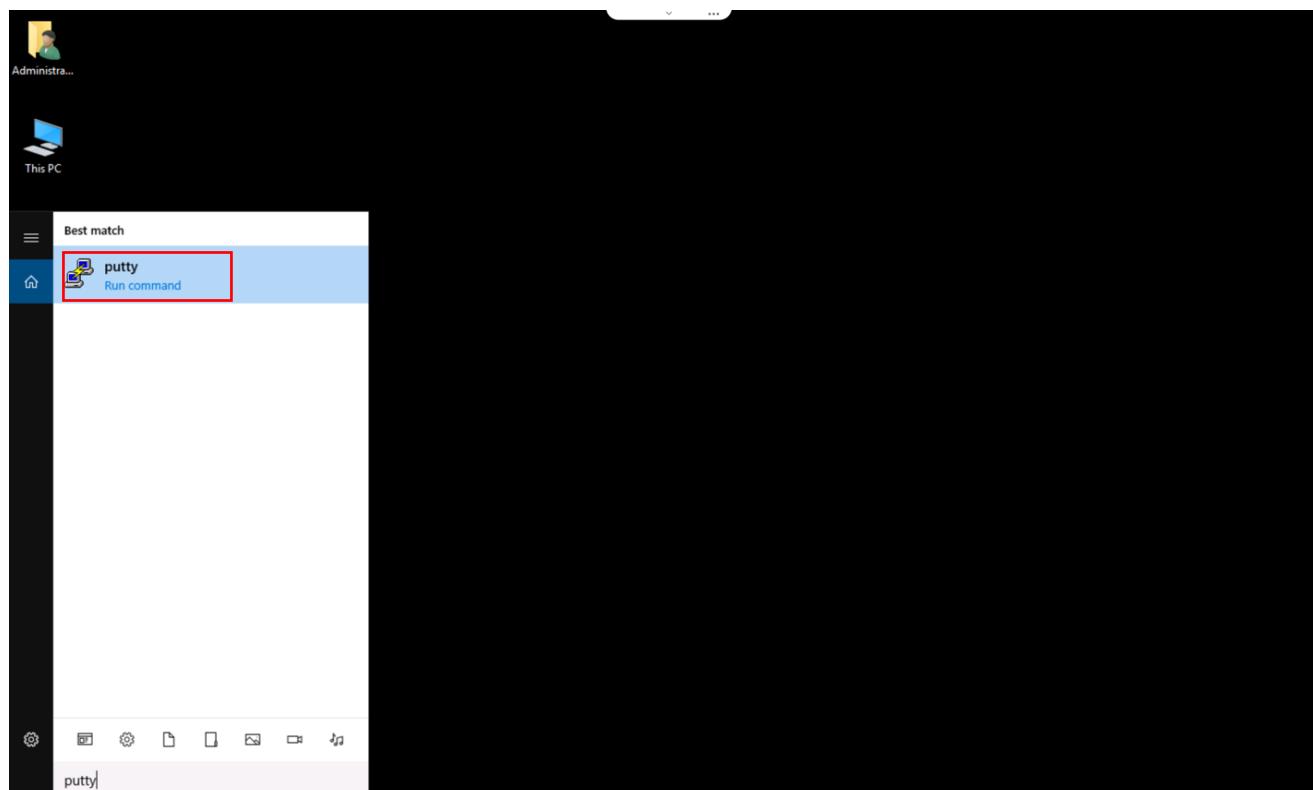
[Integrate AI model into
Triton Inference Server](#)

[Deploy Triton Inference
Server](#)

[Run sample test](#)

Access sample code

1. Open PuTTY application



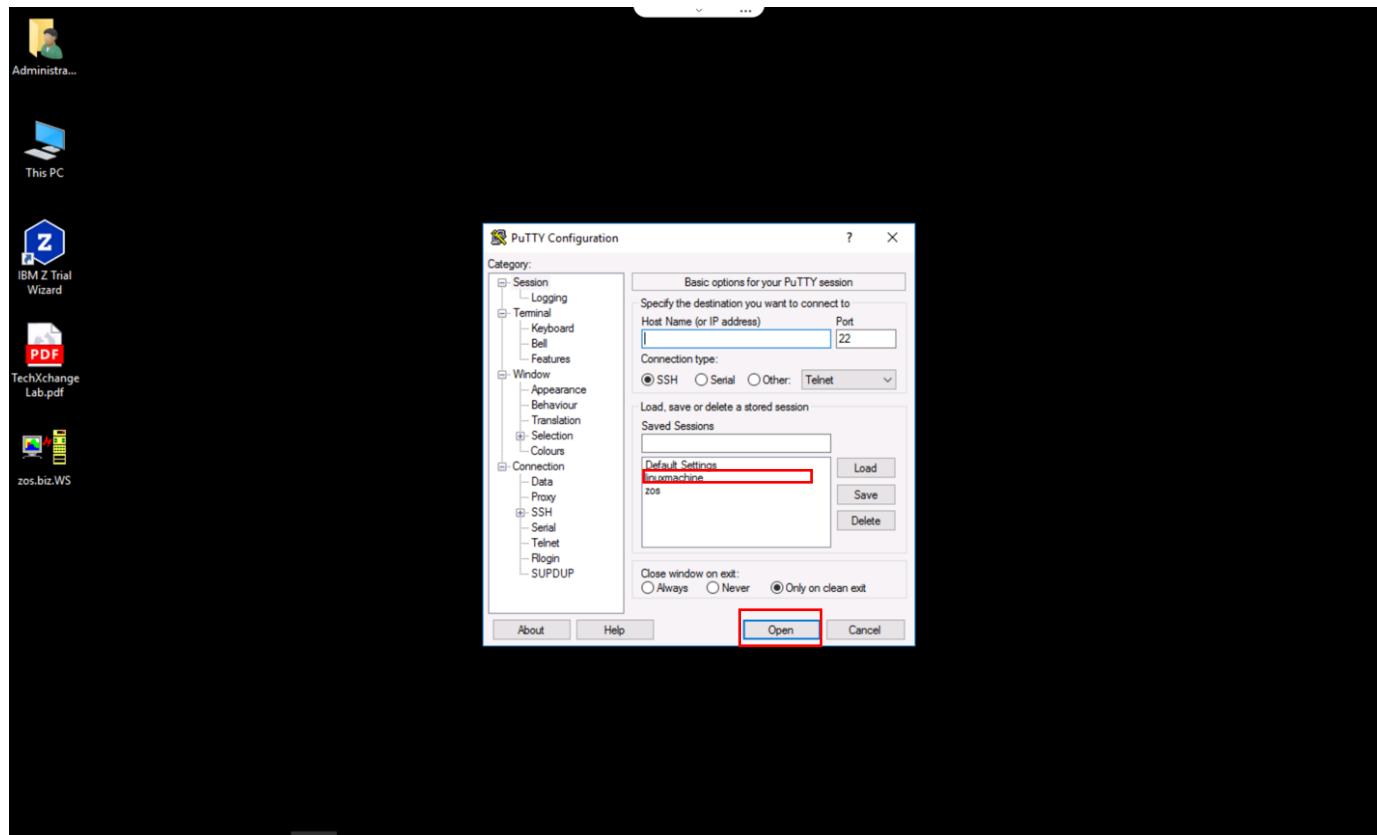
1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

2. Double click on **linuxmachine**



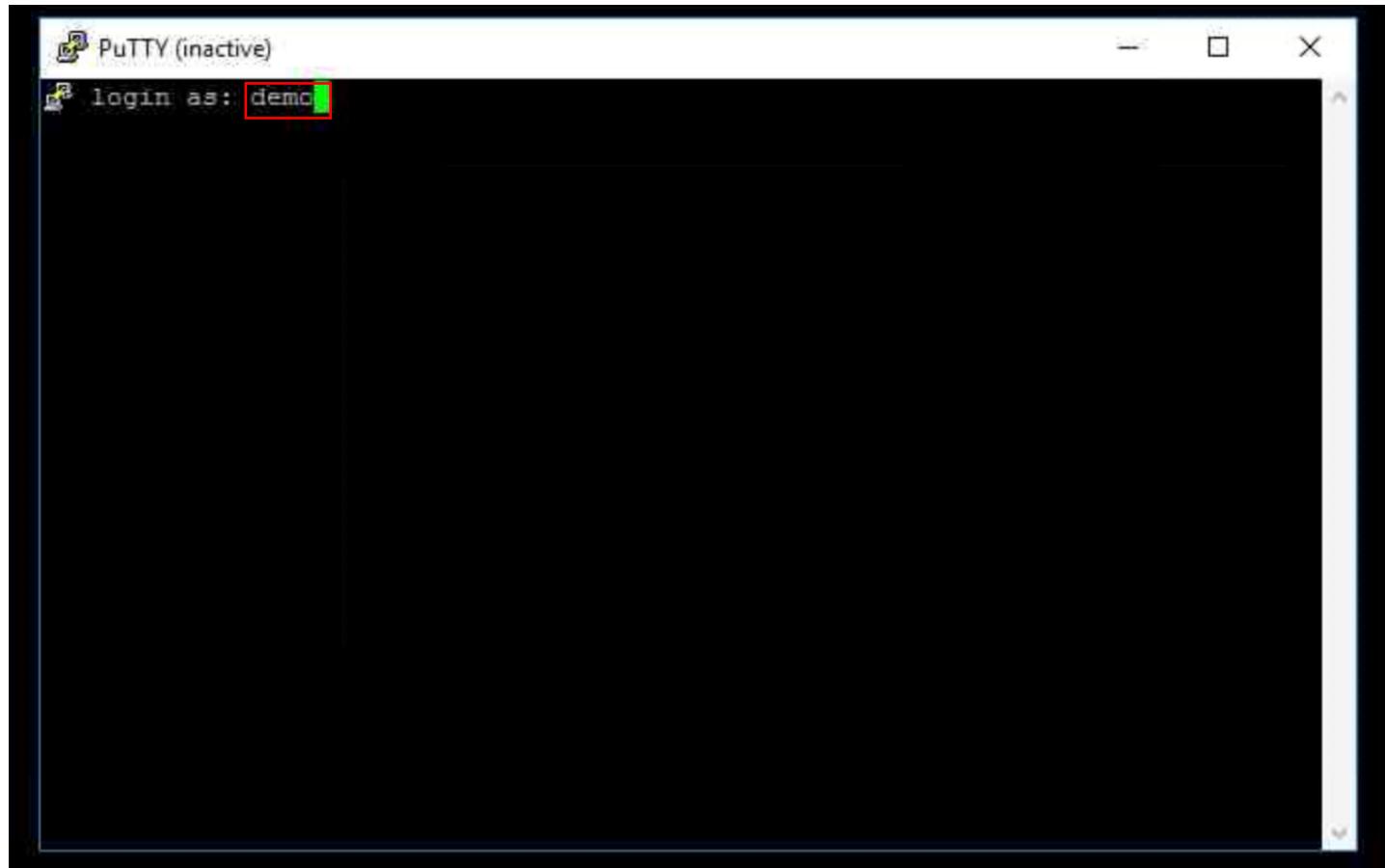
1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

3. Enter demo username



1. AI model training.

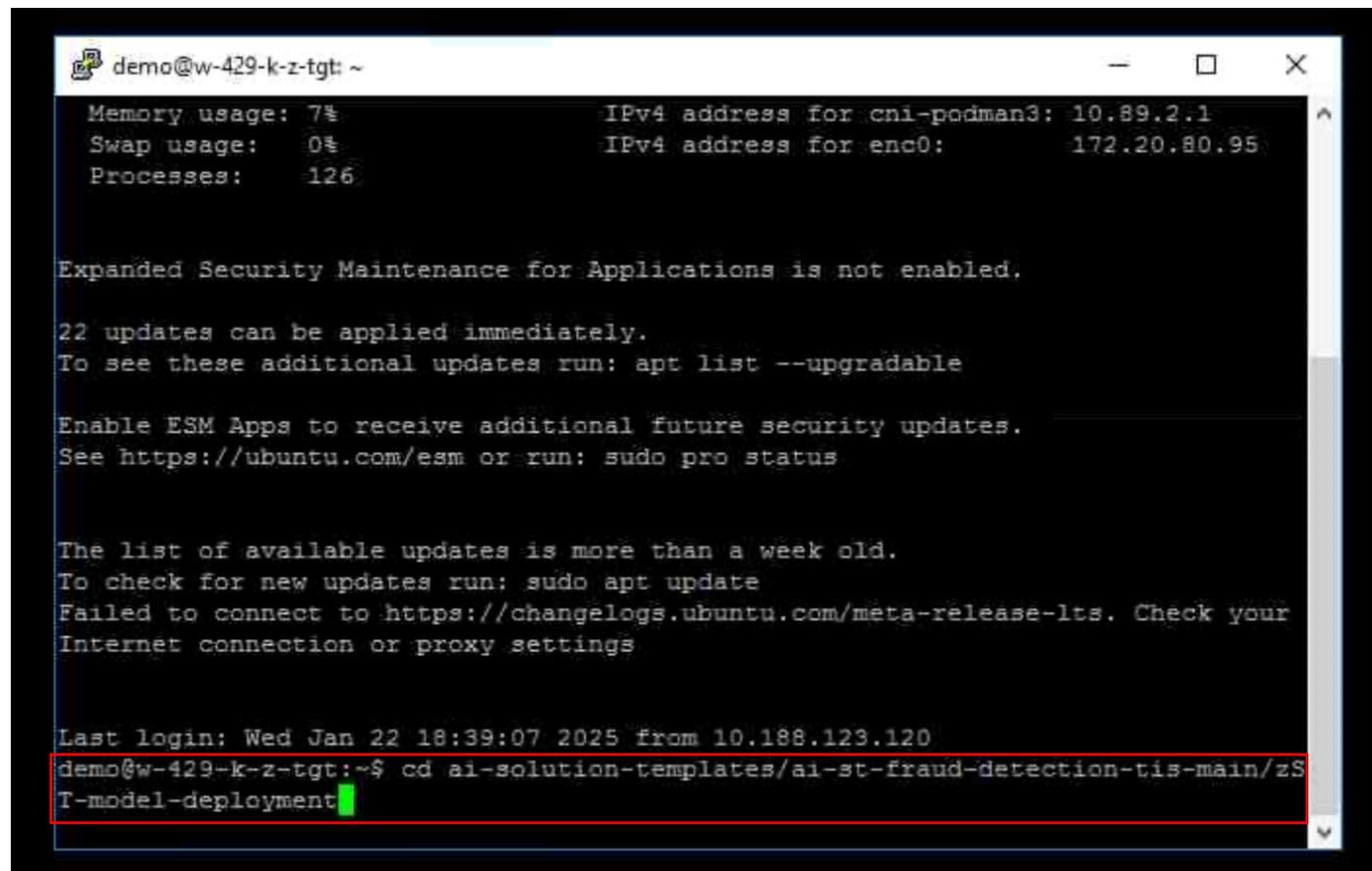
2. AI model deployment

3. AI model analysis

4. AI model integration

4. Move to sample code directory

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



A screenshot of a terminal window titled "demo@w-429-k-z-tgt: ~". The window displays various system statistics and update information. At the bottom, a command is being typed: "cd ai-solution-templates/ai-st-fraud-detection-tis-main/zS T-model-deployment". The last part of the command, "T-model-deployment", is highlighted with a red box and a green cursor.

```
demo@w-429-k-z-tgt: ~
Memory usage: 7%
Swap usage: 0%
Processes: 126
IPv4 address for cni-podman3: 10.89.2.1
IPv4 address for enc0: 172.20.80.95

Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Wed Jan 22 18:39:07 2025 from 10.188.123.120
demo@w-429-k-z-tgt:~$ cd ai-solution-templates/ai-st-fraud-detection-tis-main/zS T-model-deployment
```

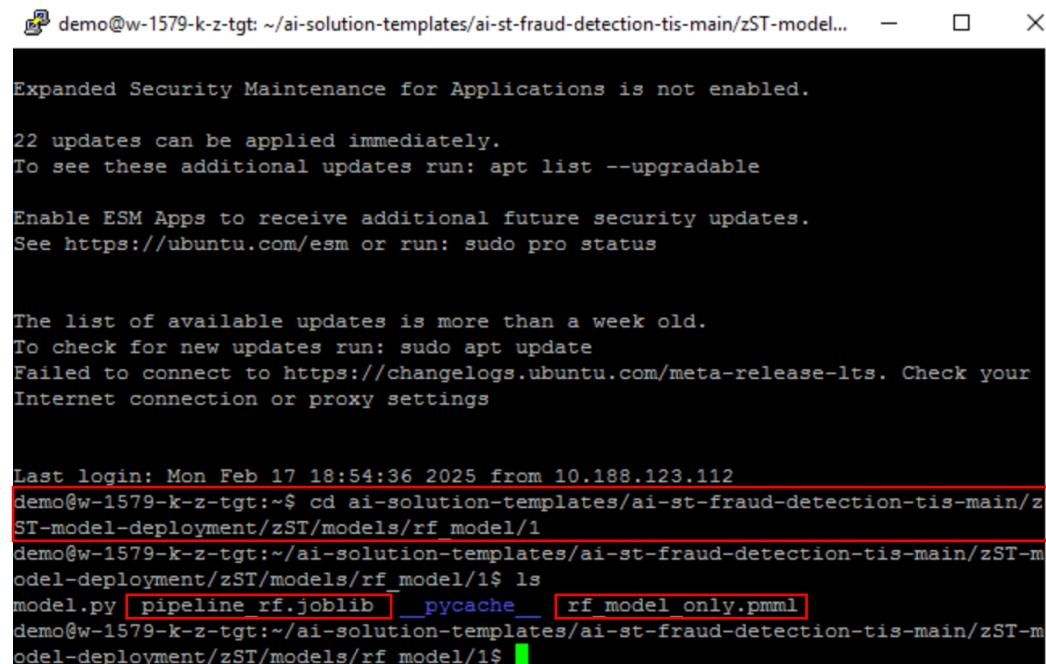
Integrate AI model into Triton Inference Server

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
```



A terminal window titled "demo@w-1579-k-z-tgt: ~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model...". The window displays several lines of text related to system updates and model deployment:

```
Expanded Security Maintenance for Applications is not enabled.  
22 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
The list of available updates is more than a week old.  
To check for new updates run: sudo apt update  
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your  
Internet connection or proxy settings  
  
Last login: Mon Feb 17 18:54:36 2025 from 10.188.123.112  
demo@w-1579-k-z-tgt:~$ cd ai-solution-templates/ai-st-fraud-detection-tis-main/z
ST-model-deployment/zST/models/rf_model/1
demo@w-1579-k-z-tgt:~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-m
odel-deployment/zST/models/rf_model/1$ ls
model.py pipeline_rf.joblib __pycache__ rf_model_only.pmml
demo@w-1579-k-z-tgt:~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-m
odel-deployment/zST/models/rf_model/1$
```

[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
```

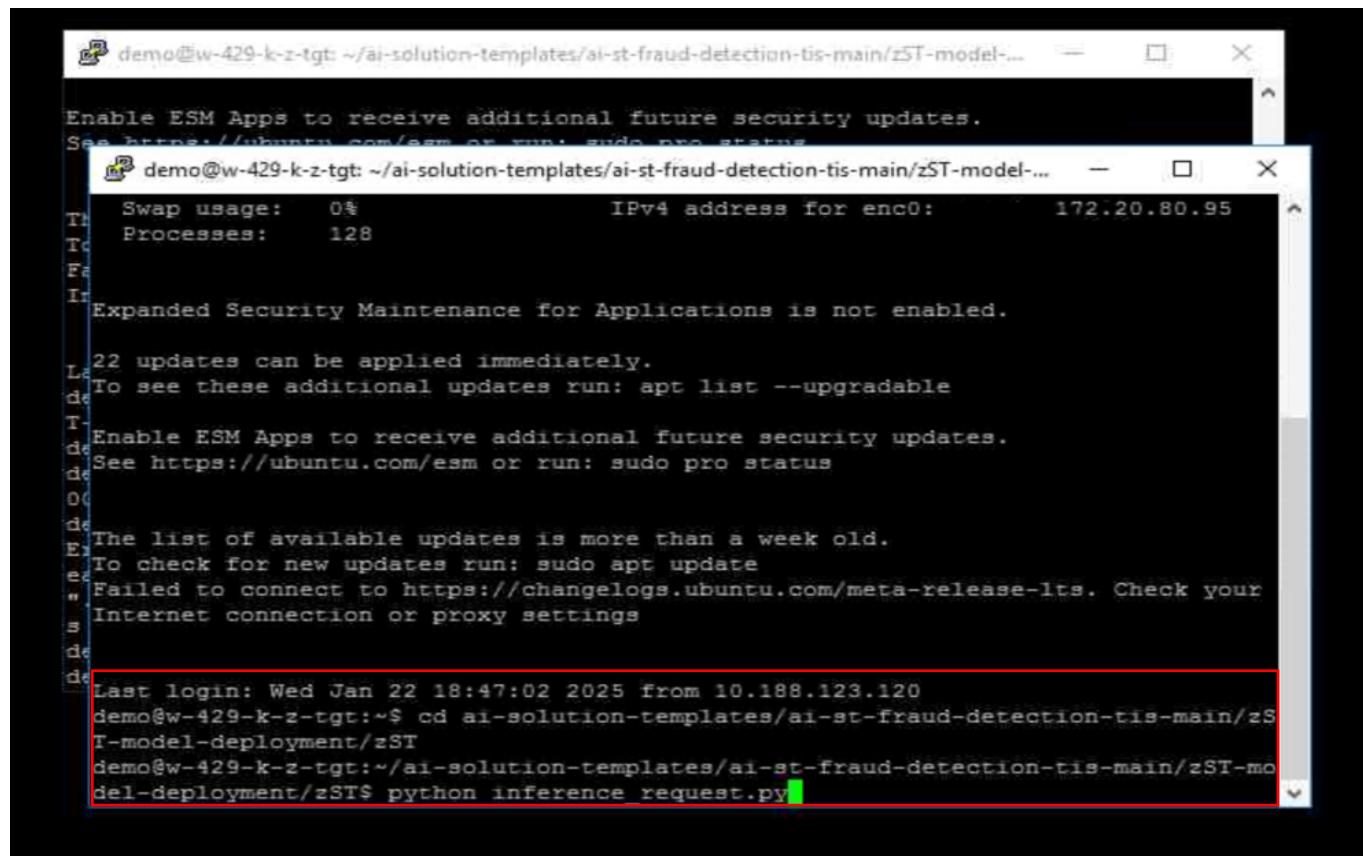
```
demo@w-429-k-z-tgt:~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-deployment$ podman run --net=my-data-network --shm-size 1G -u root --rm -p8000:8000 --name=zst-tis-app -v//$PWD/zST/models:/models zst-tis tritonserver --model-repository=/models
I0122 18:34:04.389045 1 model_lifecycle.cc:459] loading: rf_model:1
I0122 18:34:10.304987 1 python_be.cc:1977] TRITONBACKEND_ModelInstanceInitialize: rf_model_0_0 (CPU device 0)
I0122 18:34:11.379244 1 python_be.cc:1977] TRITONBACKEND_ModelInstanceInitialize: rf_model_0_1 (CPU device 0)
I0122 18:34:12.424050 1 model_lifecycle.cc:694] successfully loaded 'rf_model' version 1
I0122 18:34:12.424179 1 server.cc:583]
+-----+
| Repository Agent | Path |
+-----+
+-----+
I0122 18:34:12.424235 1 server.cc:610]
+-----+
+-----+
| Backend | Path | Config |
+-----+
```

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
```



```
demo@w-429-k-z-tgt:~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model-...
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Swap usage: 0%          IPv4 address for enc0: 172.20.80.95
Processes: 128
...
In
Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
T
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
O
de
The list of available updates is more than a week old.
E
To check for new updates run: sudo apt update
e
"Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
s
Internet connection or proxy settings
de
de
Last login: Wed Jan 22 18:47:02 2025 from 10.188.123.120
demo@w-429-k-z-tgt:~$ cd ai-solution-templates/ai-st-fraud-detection-tis-main/zS
I-model-deployment/zST
demo@w-429-k-z-tgt:~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-mo
del-deployment/zST$ python inference_request.py
```

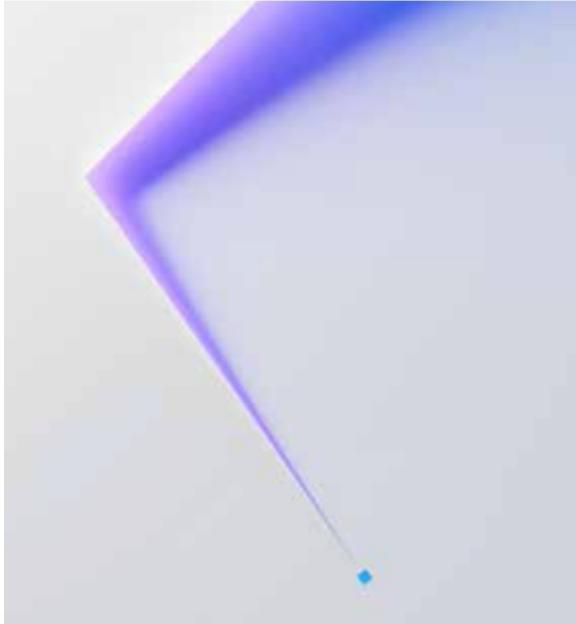
1. AI model training.

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
```

1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

[Access sample code](#)

[Configure sample application](#)

[Build sample application](#)

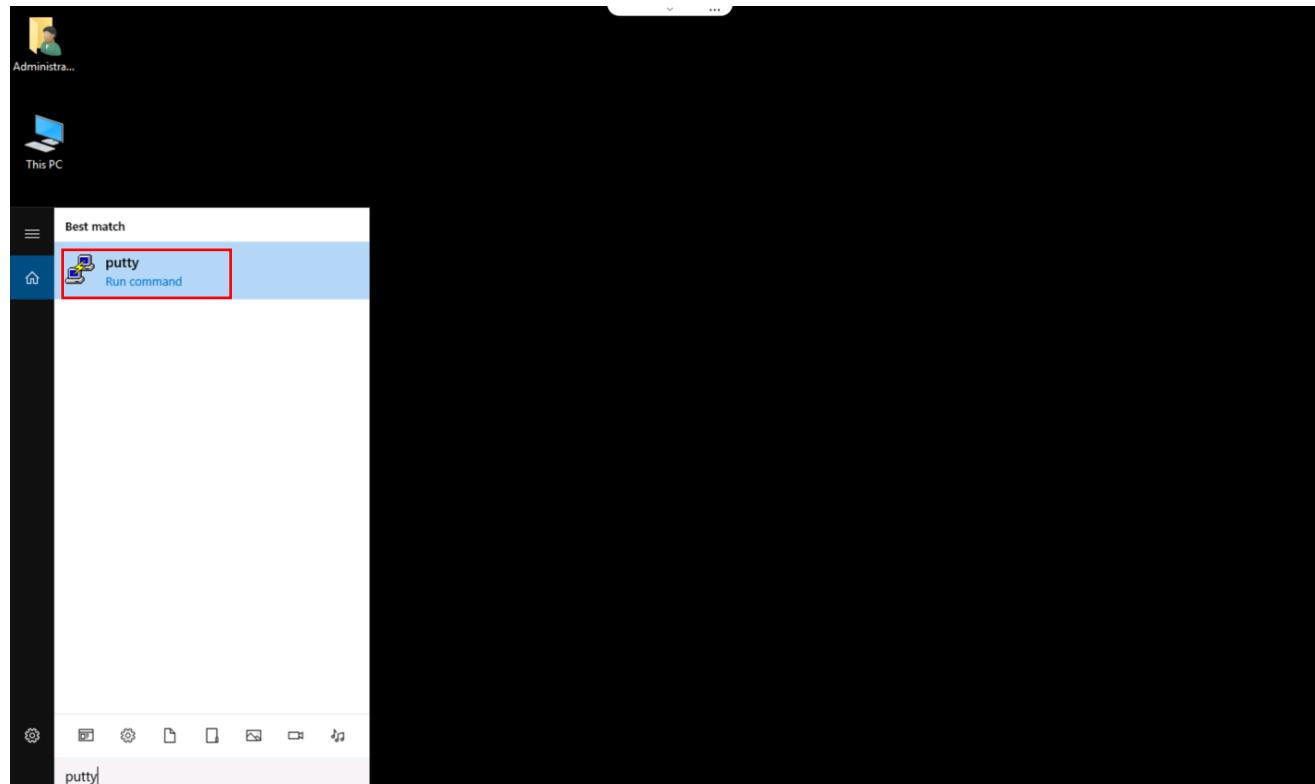
[Access sample application](#)

[Analyze credit card events](#)

[Make predication](#)

Access sample code

1. Open PuTTY application



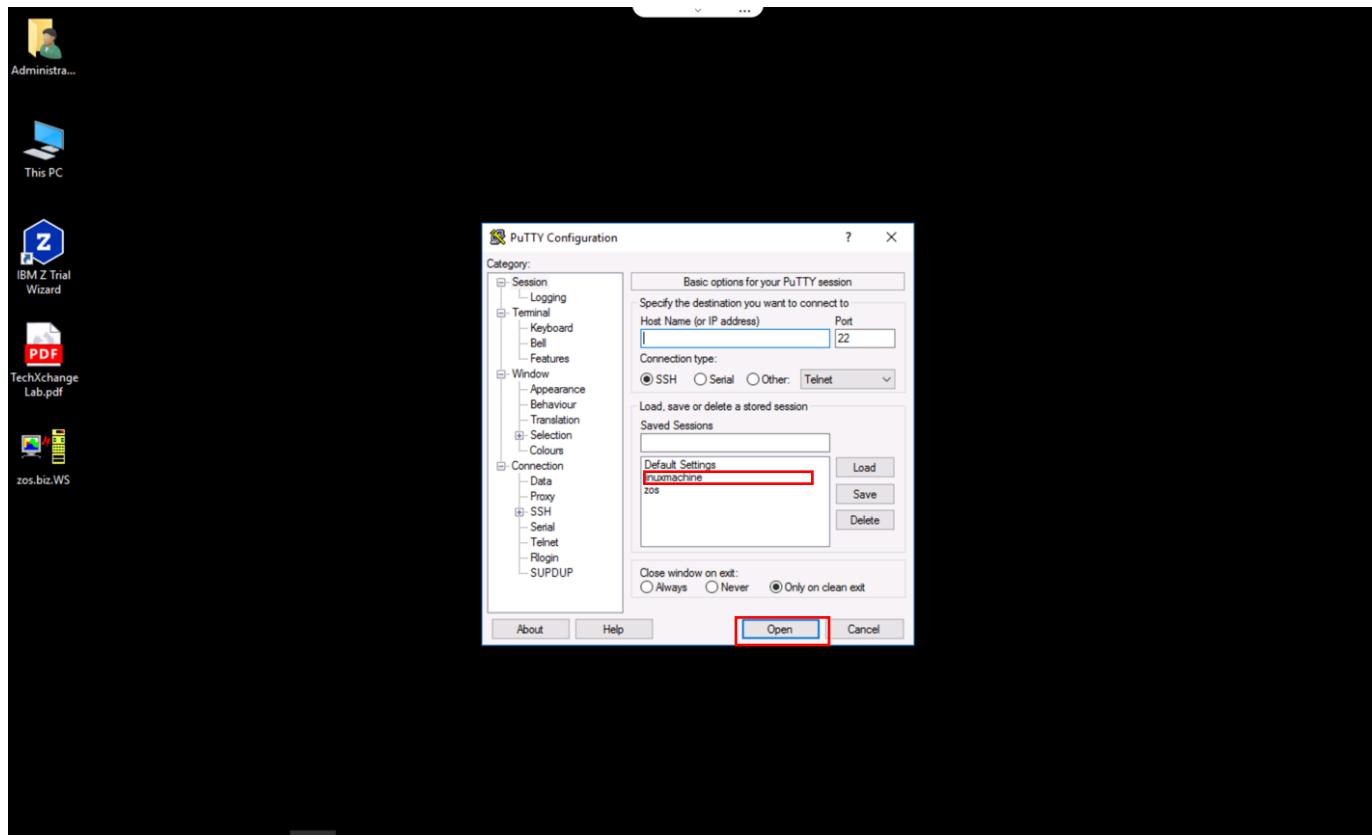
1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

2. Double click on **linuxmachine**



1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

3. Enter `demo` username



1. AI model training.

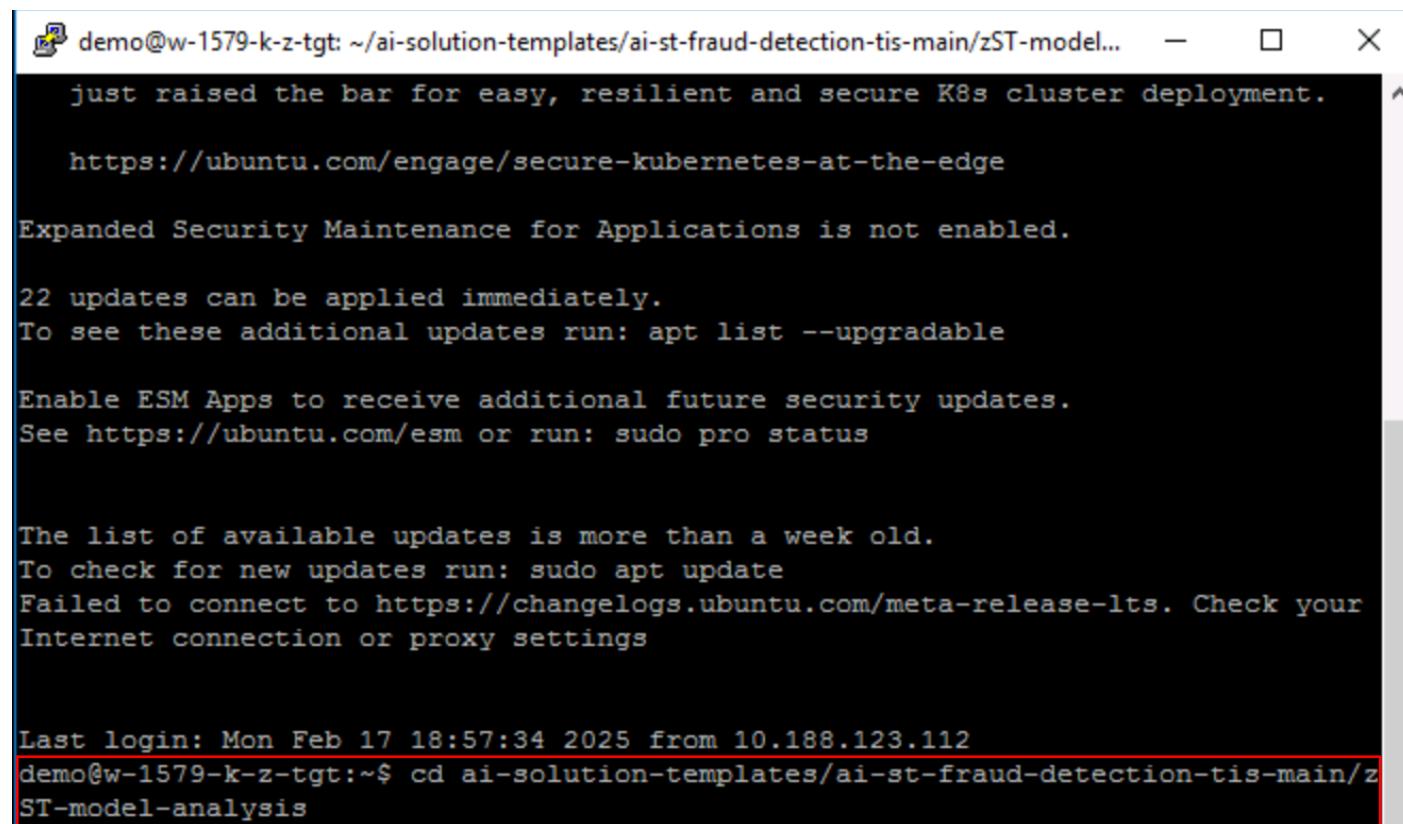
2. AI model deployment

3. AI model analysis

4. AI model integration

4. Move to sample code directory

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



A terminal window titled "demo@w-1579-k-z-tgt: ~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model..." displays the output of the "apt update" command. The output includes messages about security updates, ESM Apps, and a failed connection to the changelog server. The final line shows the user navigating back to the sample code directory.

```
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Mon Feb 17 18:57:34 2025 from 10.188.123.112
demo@w-1579-k-z-tgt:~$ cd ai-solution-templates/ai-st-fraud-detection-tis-main/z
ST-model-analysis
```

1. AI model training.

2. AI model deployment

3. AI model analysis

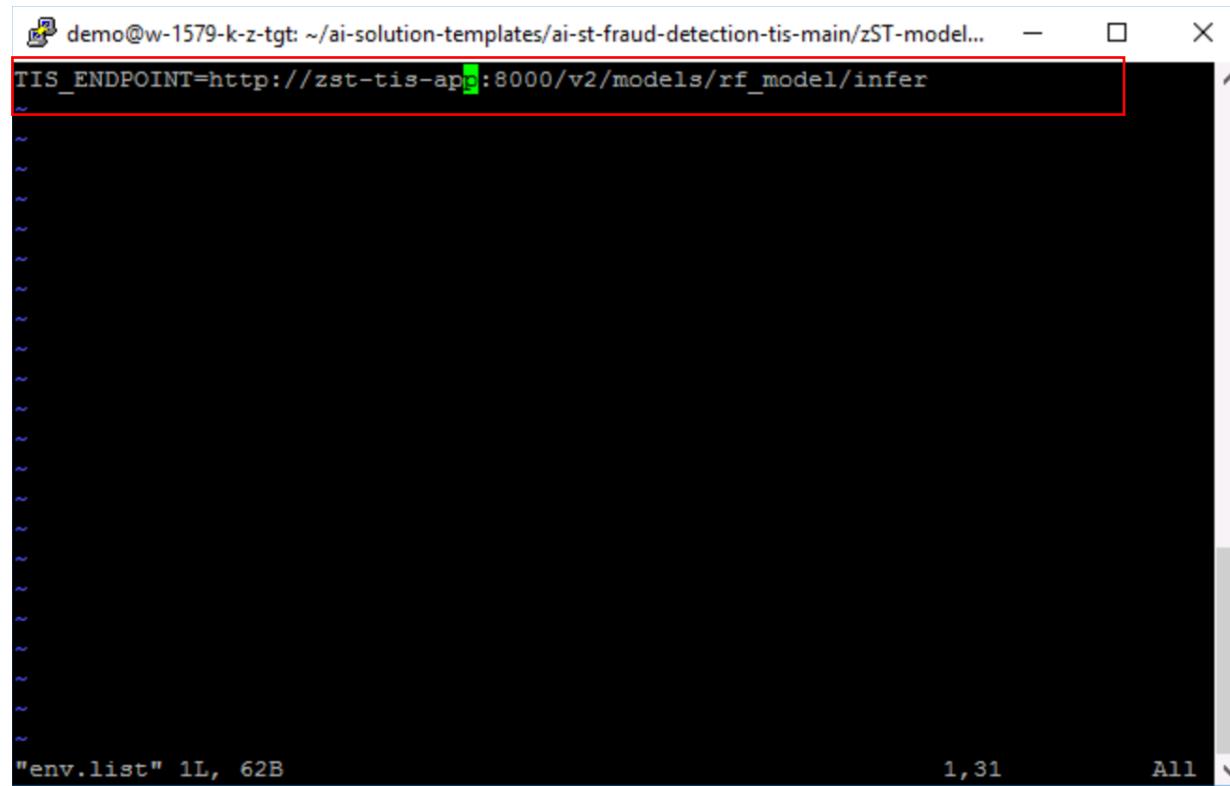
4. AI model integration

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)



A screenshot of a terminal window titled "demo@w-1579-k-z-tgt: ~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-model...". The window shows a single command being entered:

```
TIS_ENDPOINT=http://zst-tis-app:8000/v2/models/rf_model/infer
```

The terminal interface includes a status bar at the bottom showing the file path "env.list" 1L, 62B, the current line number 1,31, and the search term All.

1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

Get model details for inferencing

Access sample code

Configure sample application

[Deploy sample application](#)

[Access sample application](#)

Analyze credit card events

Make prediction

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
```

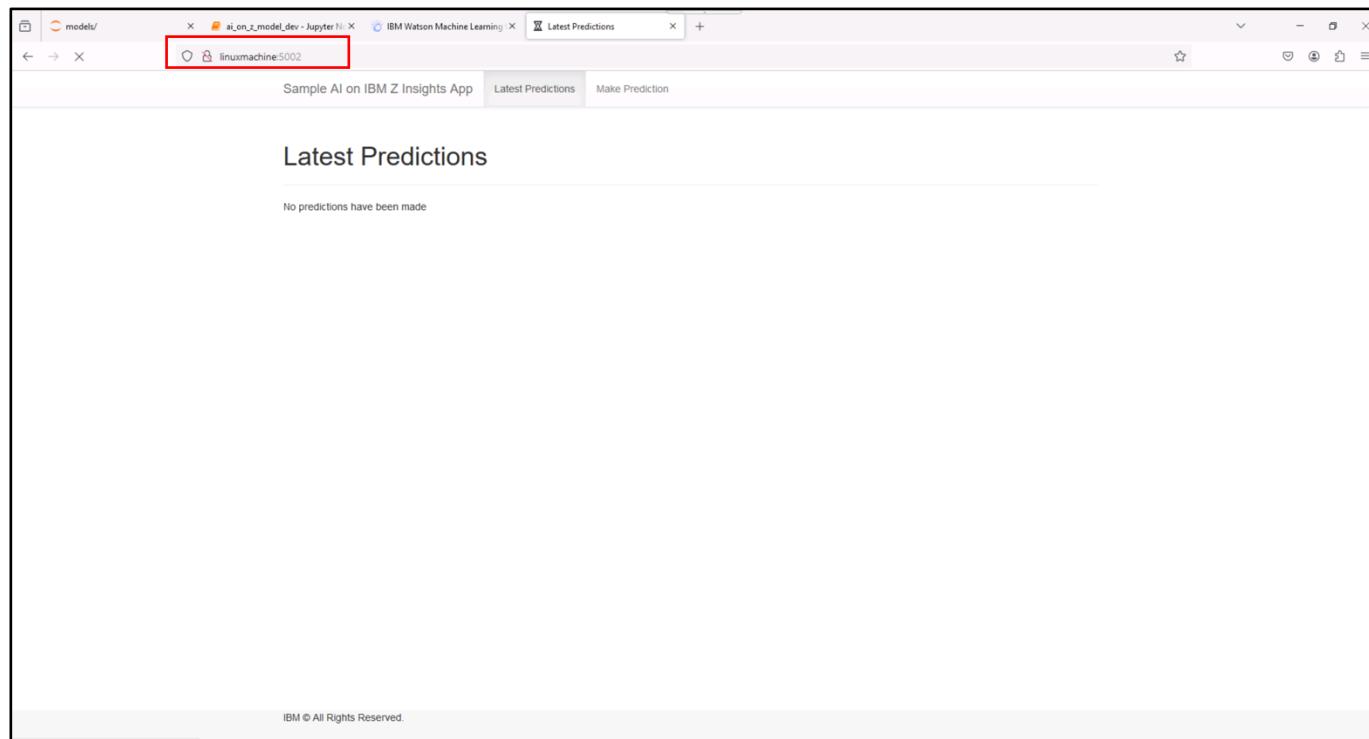
The screenshot shows a terminal window with two panes. The left pane displays configuration options for a Triton server, including fields for server_id, server_version, server_extensions, model_repository_path[0], model_control_mode, strict_model_config, rate_limit, pinned_memory_pool_byte_size, min_supported_compute_capability, strict_readiness, exit_timeout, and cache_enabled. The right pane shows the command being run: `podman run --rm -p 5002:5002 --net=my-data-network --env-file env.list --name model-analysis-tis-app model-analysis-tis`. The output of the command indicates that the Flask app is running on port 5002, and it also lists other ports (127.0.0.1:5002 and 10.89.1.3:5002) and provides instructions to press CTRL+C to quit. A red box highlights the command line in the right pane.

Access sample application

1. View the following URL in a web browser

<http://linuxMachine:{port}>

- a. Port: port you used with podman run



1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

Get model details for
inferencing

Access sample code

Configure sample
application

Deploy sample application

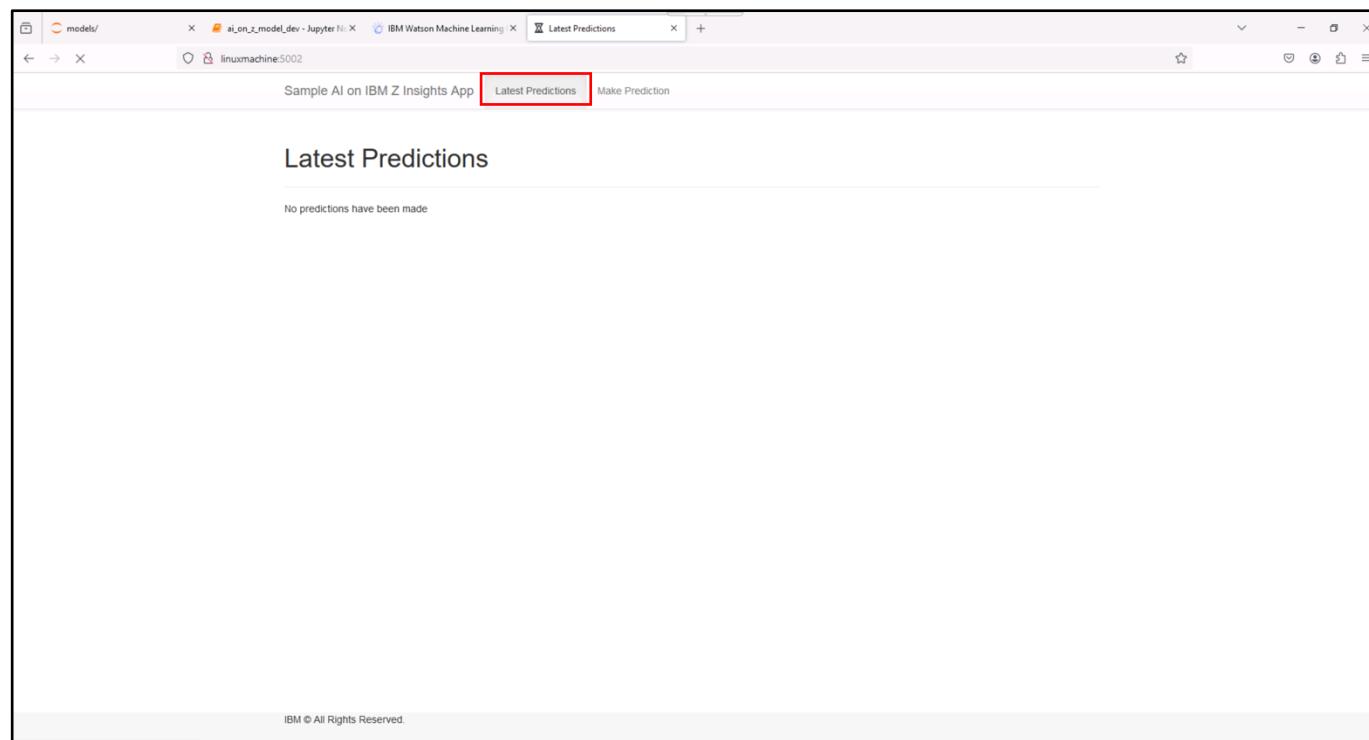
Access sample application

Analyze credit card events

Make prediction

Analyze credit card events

1. Go to latest predictions tab



1. AI model training.

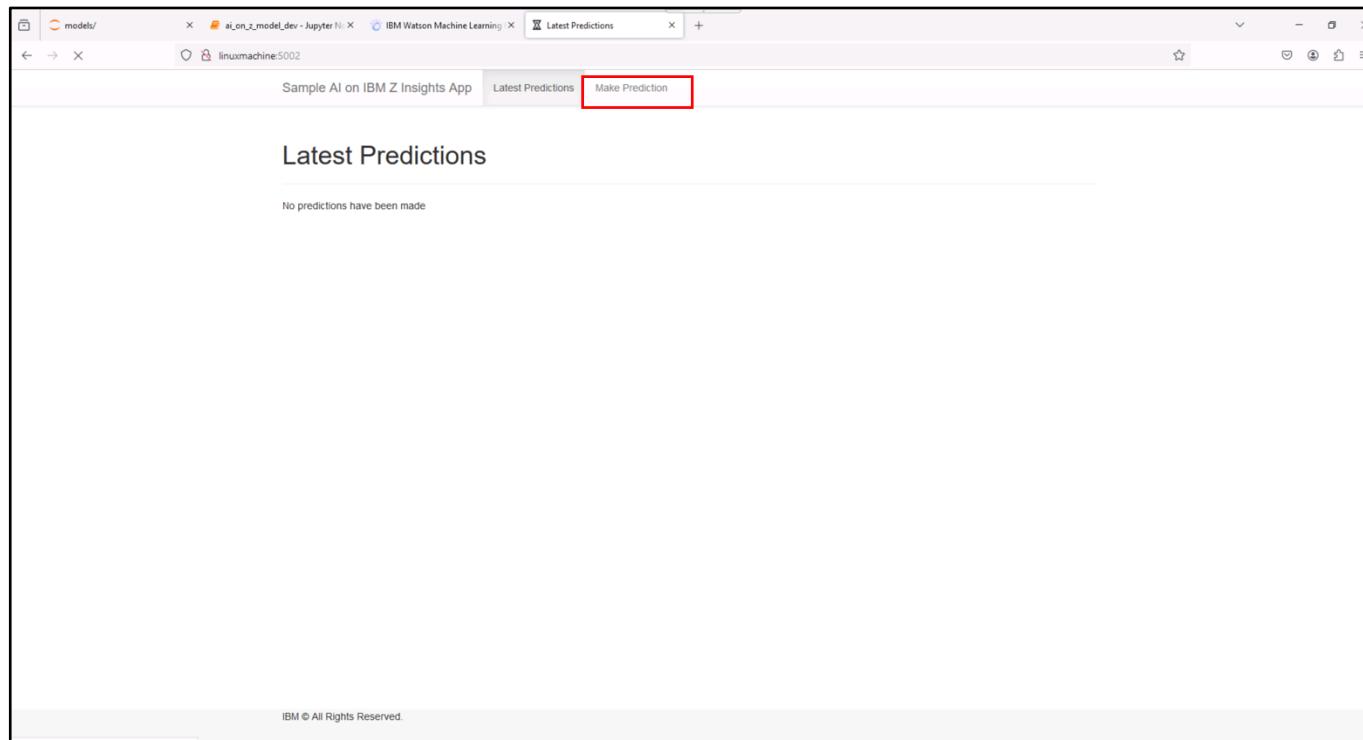
2. AI model deployment

3. AI model analysis

4. AI model integration

Make predication

1. Go to latest make predication tab



1. AI model training.

2. AI model deployment

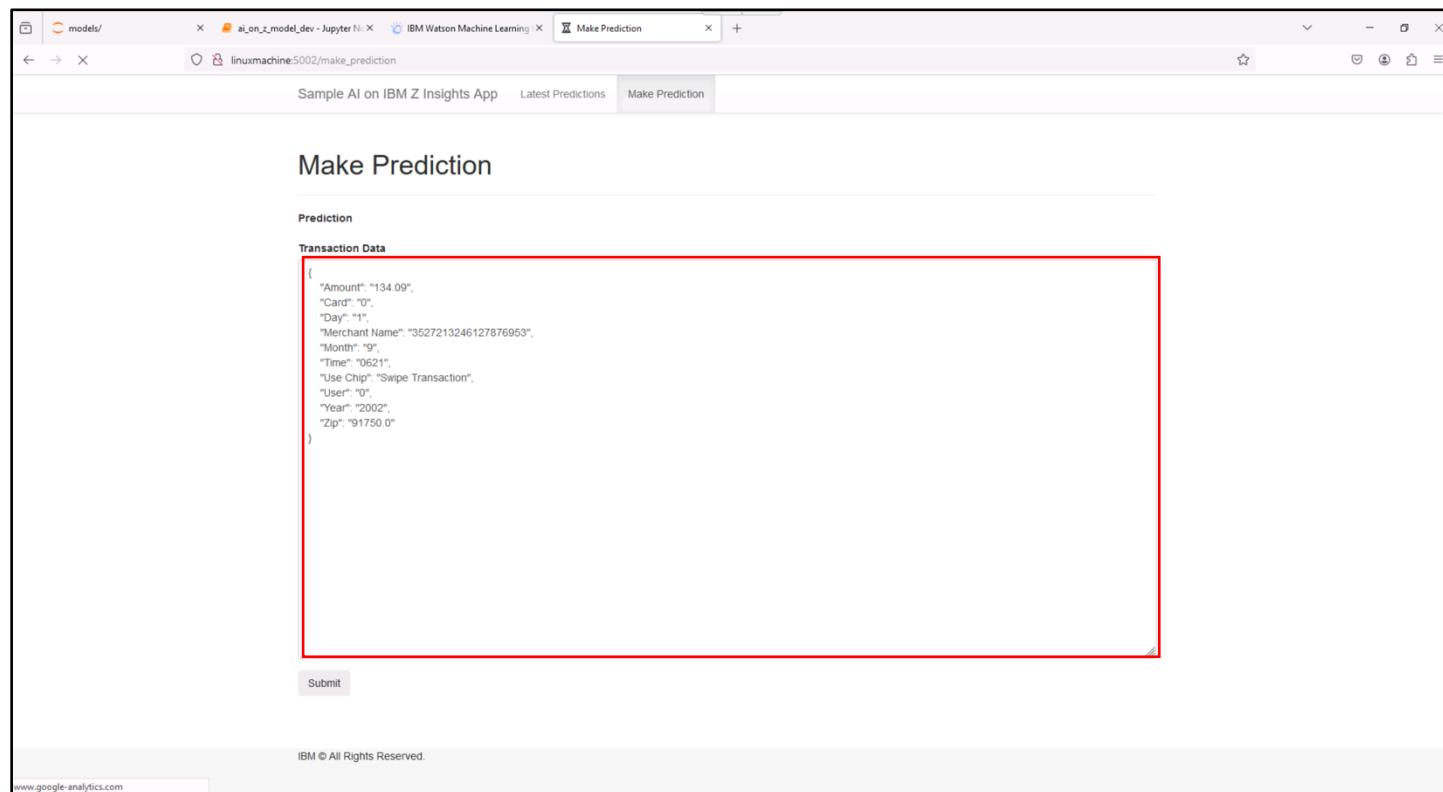
3. AI model analysis

4. AI model integration

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"}
```



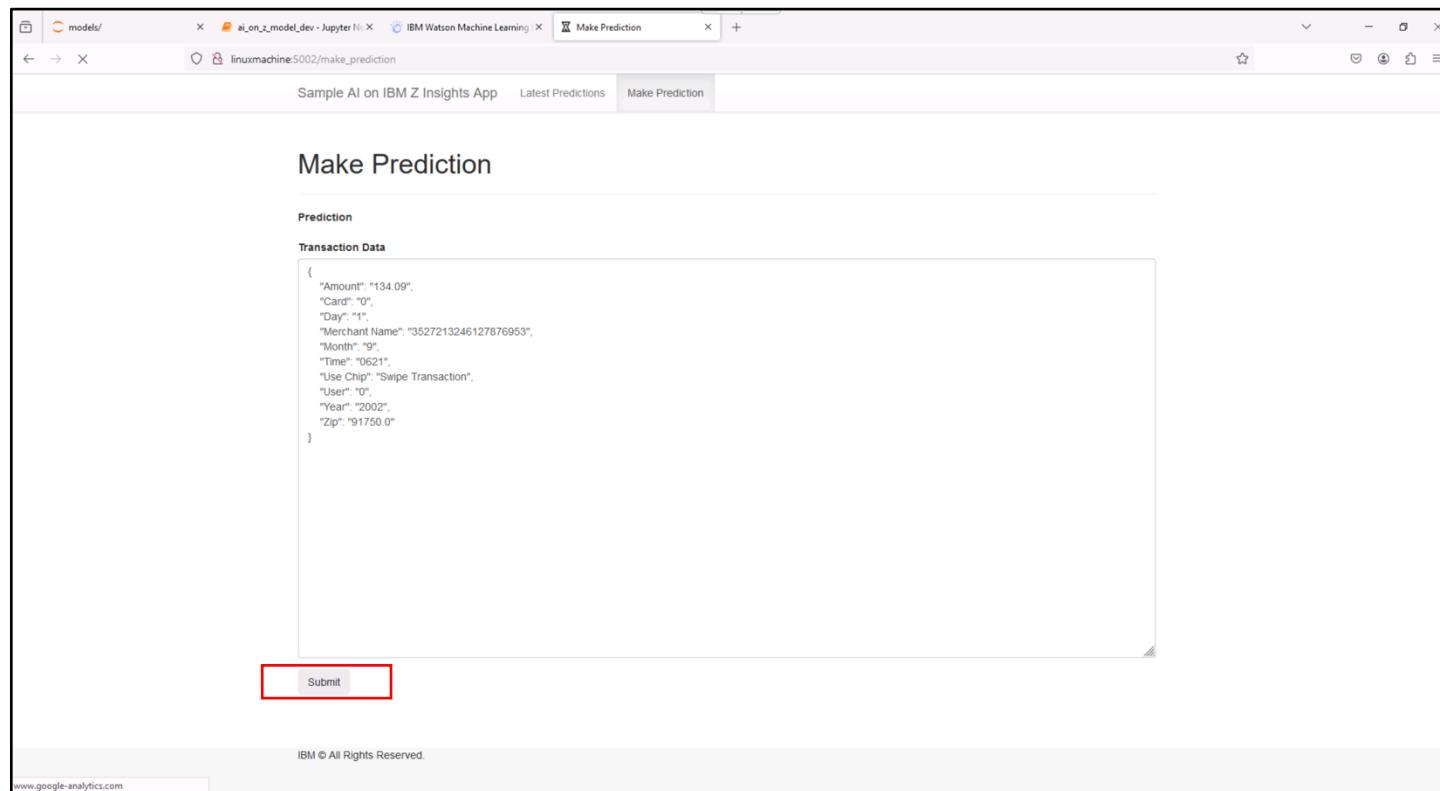
1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

3. Click submit



1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

AI model analysis complete



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

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
```

1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

[Access sample code](#)

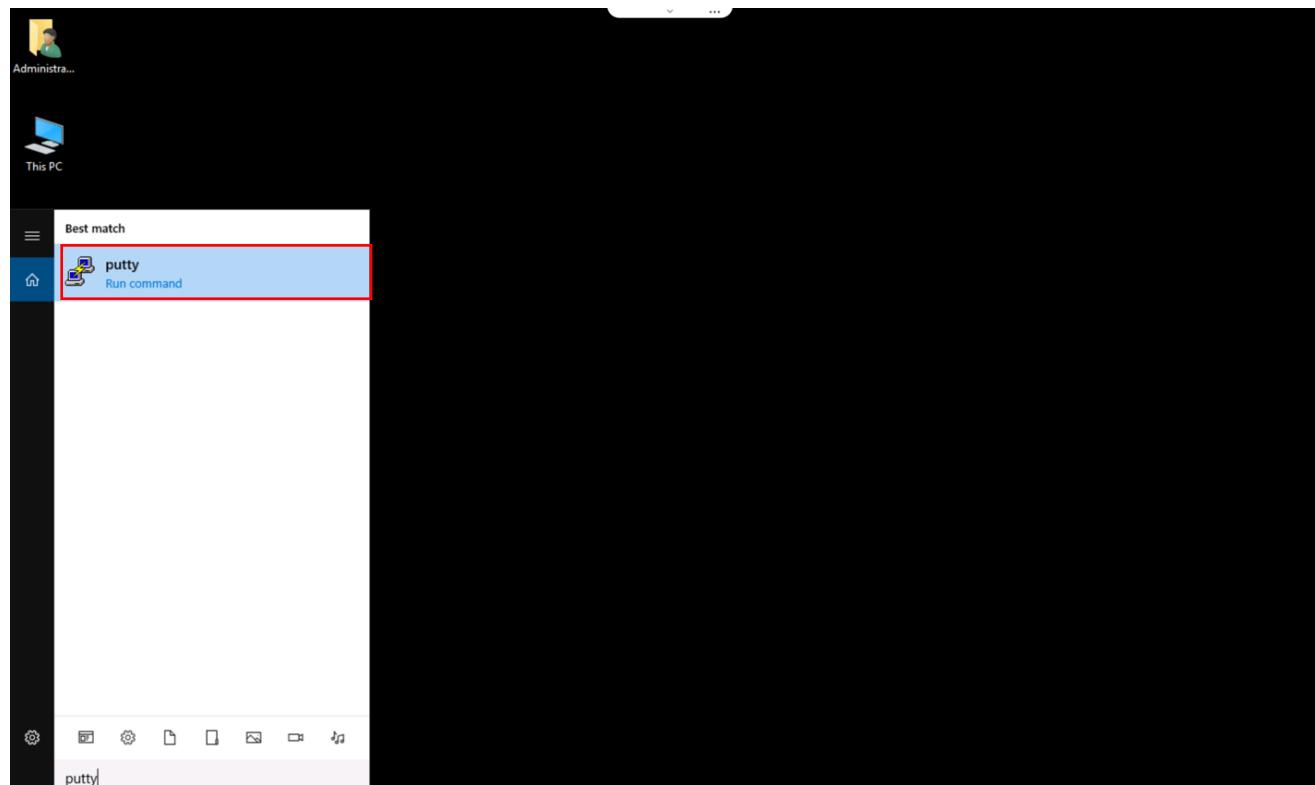
[Deploy sample ecommerce application](#)

[Access sample ecommerce application](#)

[Use fraud detection AI model with EverShop Storefront](#)

Access sample code

1. Open PuTTY application



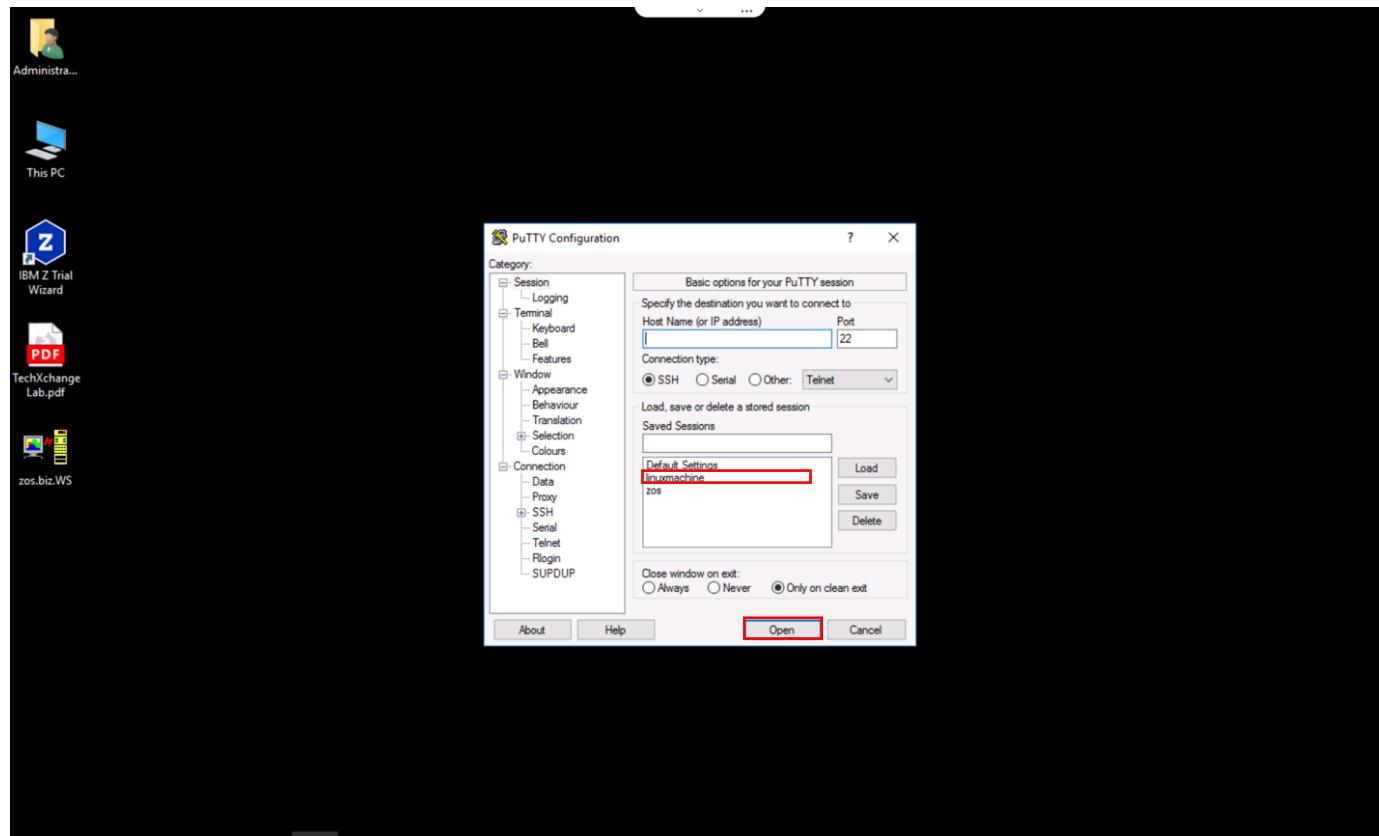
1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

2. Double click on **linuxmachine**



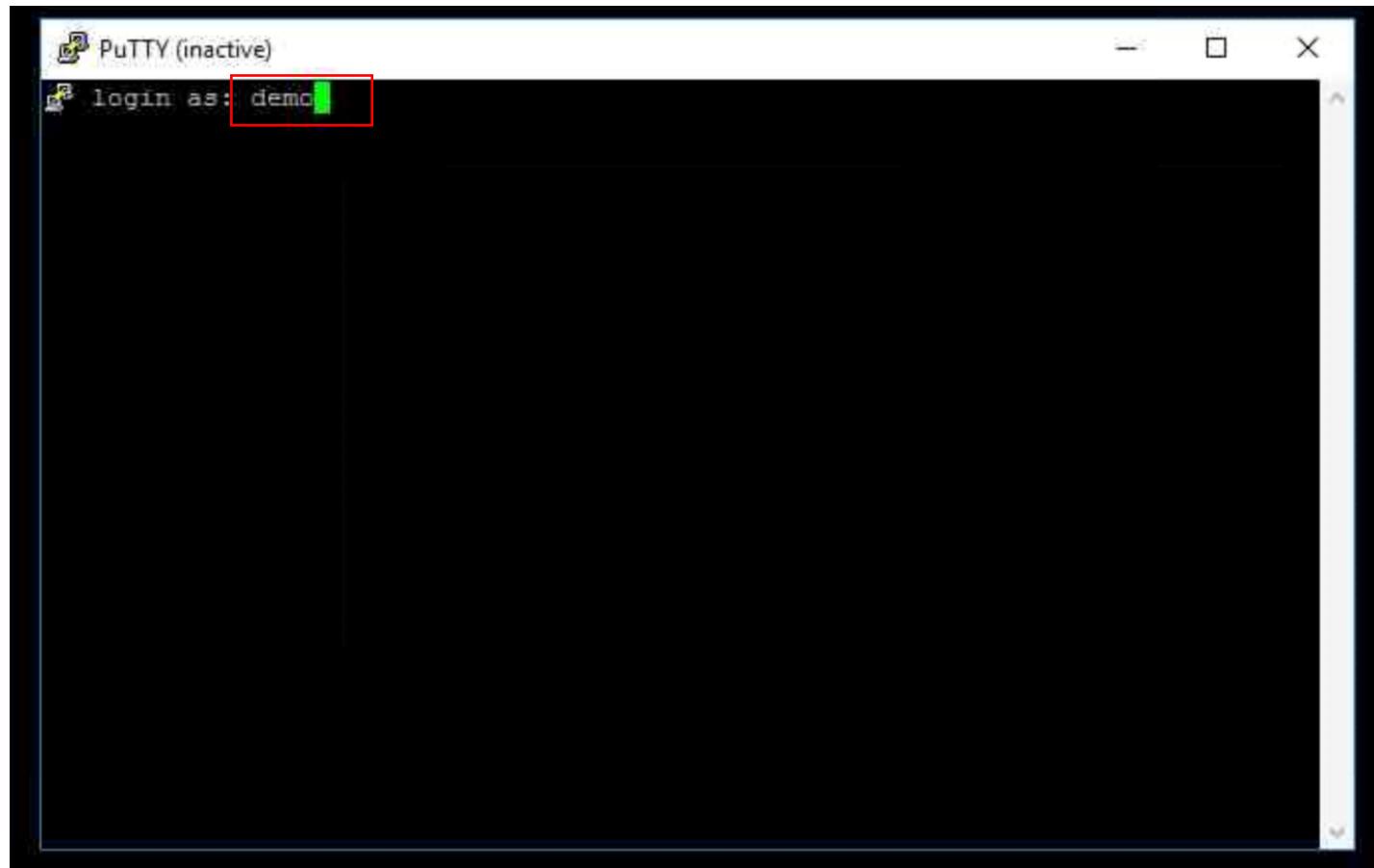
1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

3. Enter demo username



1. AI model training.

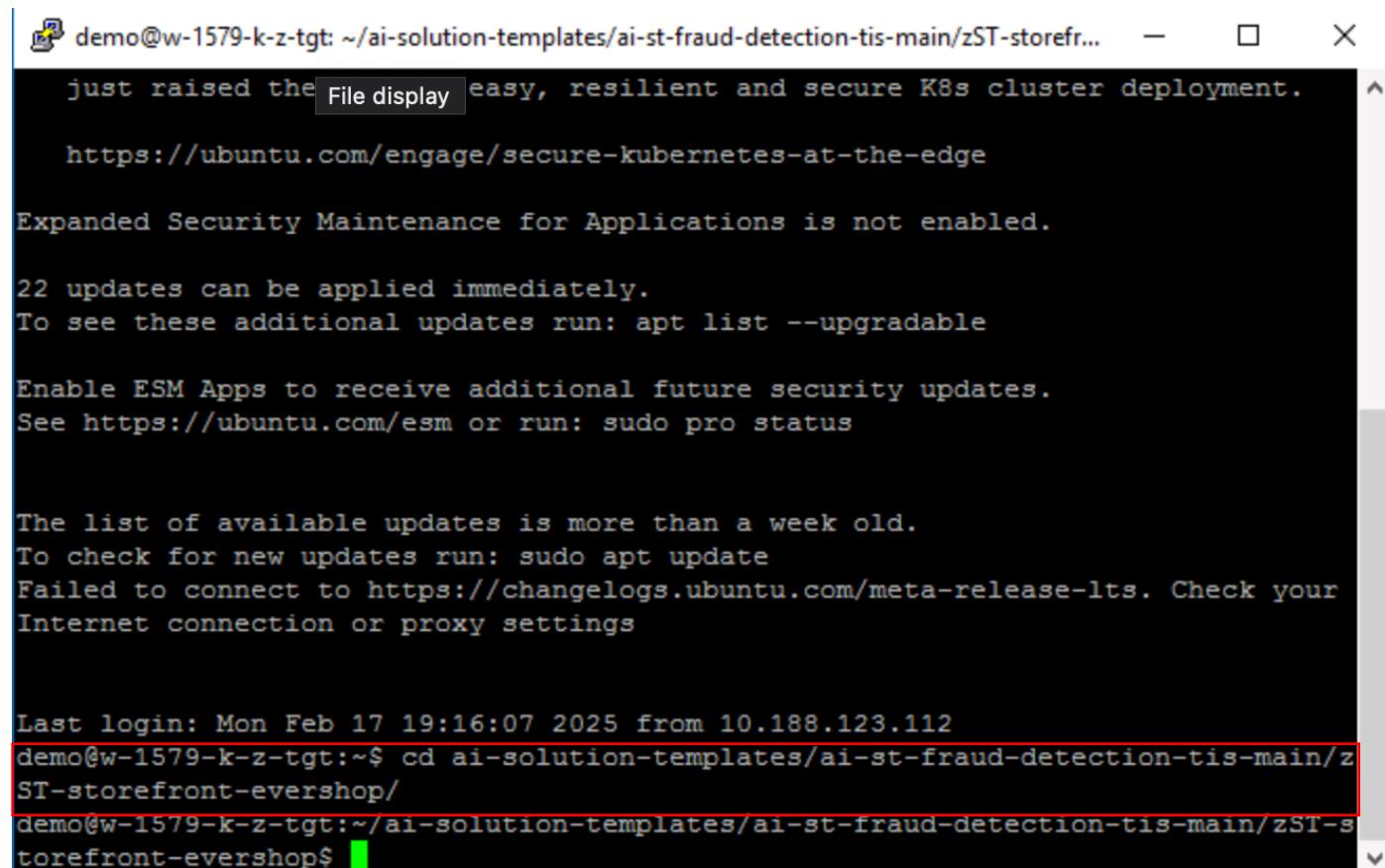
2. AI model deployment

3. AI model analysis

4. AI model integration

4. Move to sample code directory

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



A terminal window titled "demo@w-1579-k-z-tgt: ~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-storefr...". The window displays the output of a command to change directory to the specified path. The output includes system status information, update availability, and a warning about old package lists. The final line shows the command being run again, with the path highlighted by a red rectangle.

```
just raised the File display easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your
Internet connection or proxy settings

Last login: Mon Feb 17 19:16:07 2025 from 10.188.123.112
demo@w-1579-k-z-tgt:~/ai-solution-templates/ai-st-fraud-detection-tis-main/z
ST-storefront-evershop/
demo@w-1579-k-z-tgt:~/ai-solution-templates/ai-st-fraud-detection-tis-main/zST-s
torefront-evershop$
```

1. AI model training.

2. AI model deployment

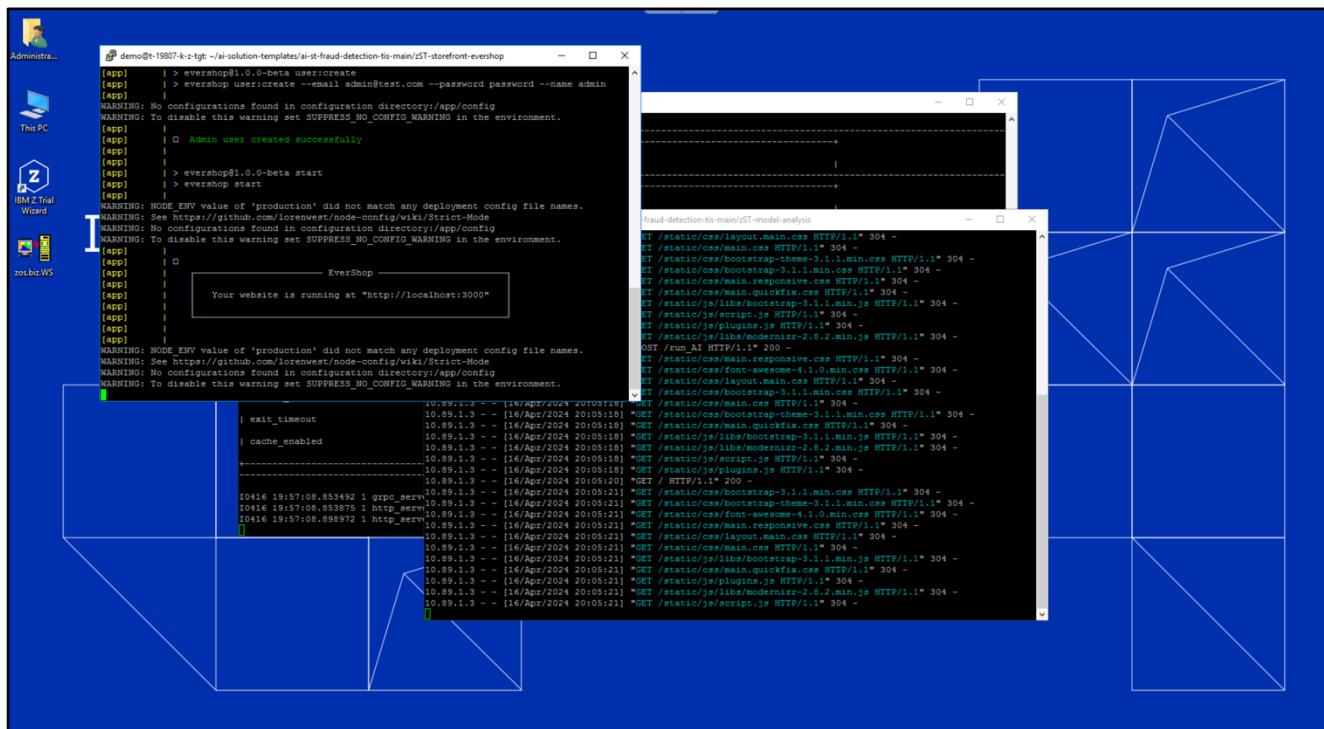
3. AI model analysis

4. AI model integration

Deploy sample ecommerce application

1. Run command in terminal (password: demo)

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



1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

Access sample code

Deploy sample
ecommerce application

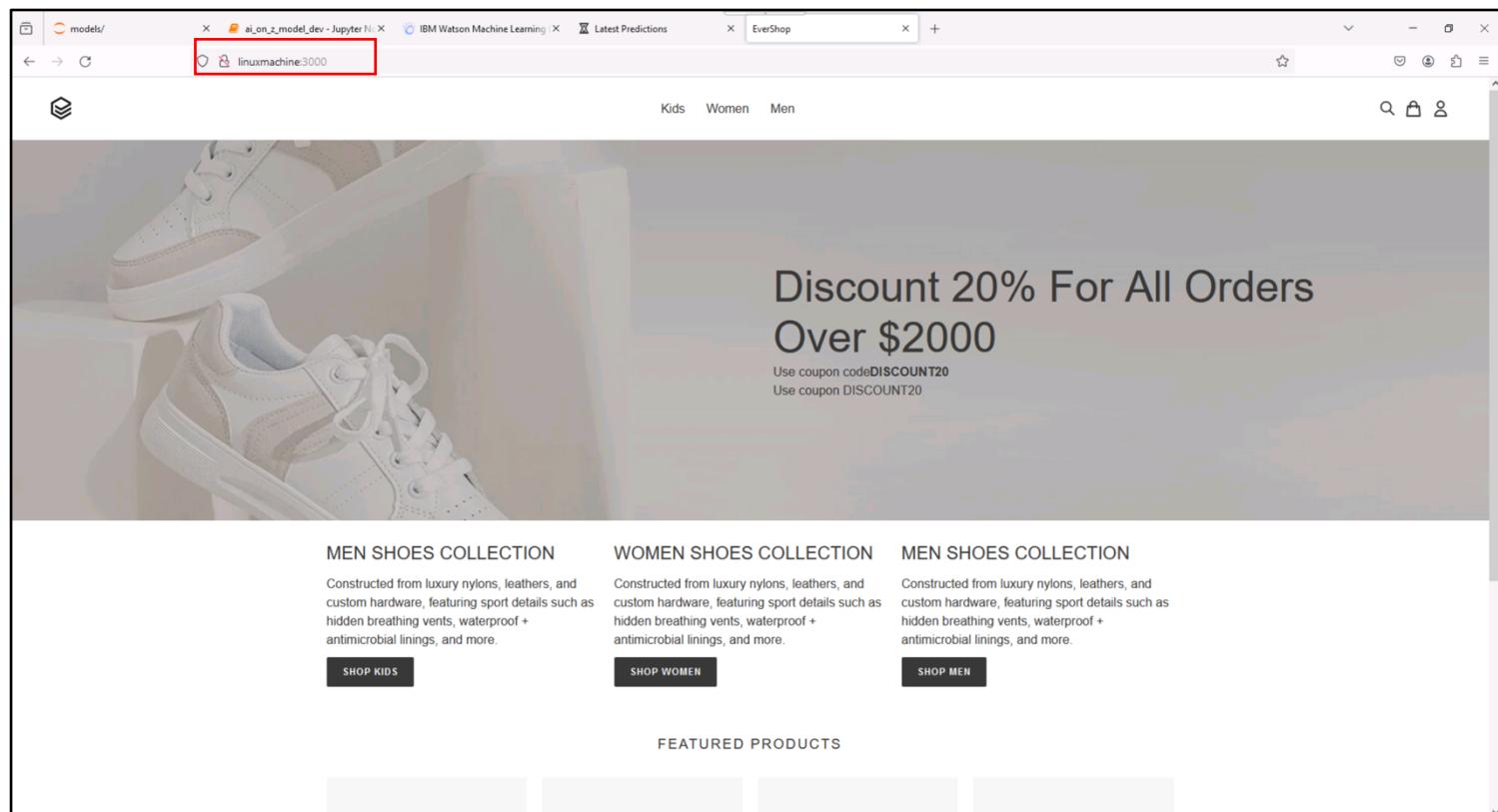
Access sample
ecommerce application

Use fraud detection AI
model with EverShop
Storefront

Access sample ecommerce application

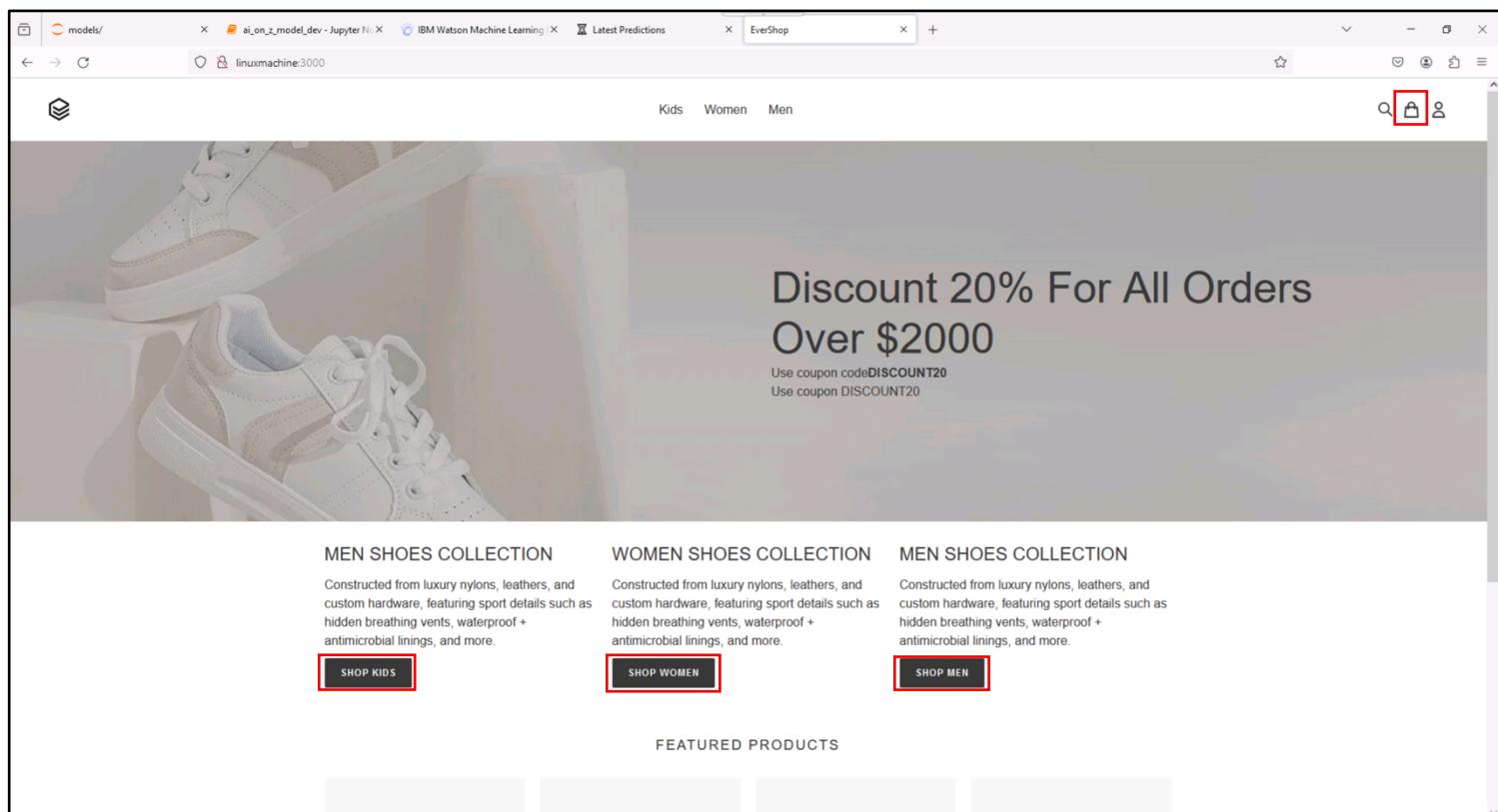
1. Enter URL in web browser using app URL

<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



1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

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

The screenshot shows a web browser window with the following details:

- Address bar: models/ - ai_on_z_model_dev - Jupyter Notebook - IBM Watson Machine Learning - Latest Predictions - Checkout
- Page title: Checkout
- Header: Kids Women Men
- Breadcrumbs: Home / Checkout
- Form fields:
 - Contact: test@test.com
 - Ship to: test, test, United States
- Billing Address section:
 - My billing address is same as shipping address (checked)
- Payment Method section:
 - Card icons: American Express, MasterCard, Visa
 - Test failure message box:
 - Test failure:
 - Test card number: 4000 0000 0000 9995
 - Test card expiry: 04/24
 - Test card CVC: 242
 - Buttons: Powered by stripe, Test success, Test failure
- Order summary:

1	Striped Cotton Sweater	\$90.00
Sub total	1 items	\$90.00
Shipping	FedEx	\$10.00
Discount		\$0.00
Total		\$100.00
(Inclusive of tax \$0.00)		
- Message: Payment failed. Please contact your credit card provider for details. Refresh page to continue.

1. AI model training.

2. AI model deployment

3. AI model analysis

4. AI model integration

AI model integration complete