AI on Linux on Z

Data preprocessing solution template

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

Table of contents

AI model training.....3

AI model deployment.....8

Preprocessing service deployment.....12

AI model inferencing.....15

Step 1

AI model training

We will build a fraud detection AI model by training with the provided rapid AI on Linux on 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 TIS.

All sample code for this section is within

`aionz-st-data-preprocessing-tis/zST-model-training-jupyter`

Prerequisites

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

Access rapid AI on Linux on Z development environment

Provide data

Model training

Access trained AI model

Access rapid AI on Linux on Z development environment

1. Access sample code

   ```
   cd zST-model-training-jupyter
   ```

2. Create and activate Python virtual environment

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

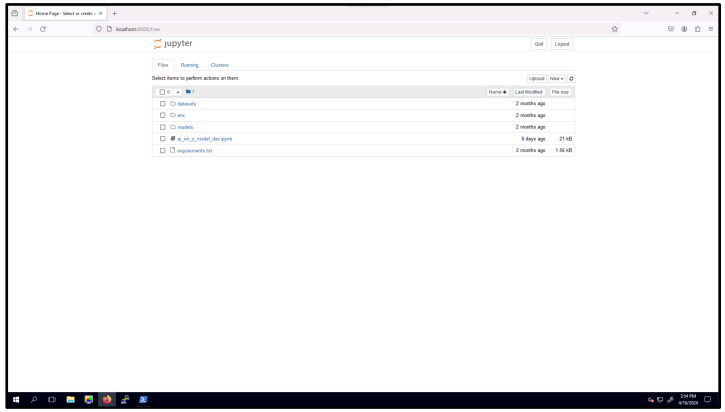
3. Install required Python packages

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

4. Run Jupyter

   ```
   jupyter notebook
   ```

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



Access rapid AI on Linux on Z development environment

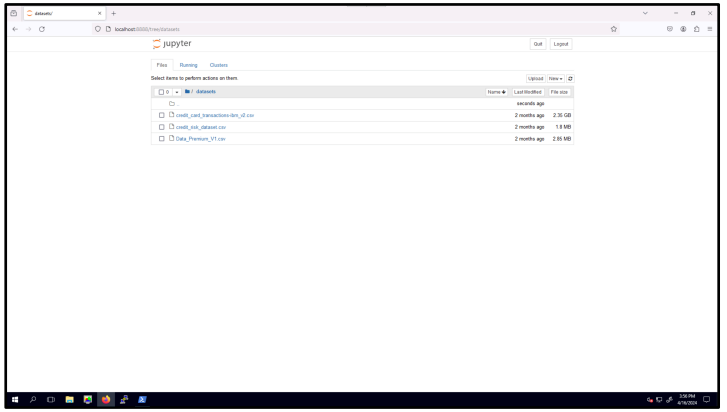
Provide data

Model training

Access trained AI model

Provide data

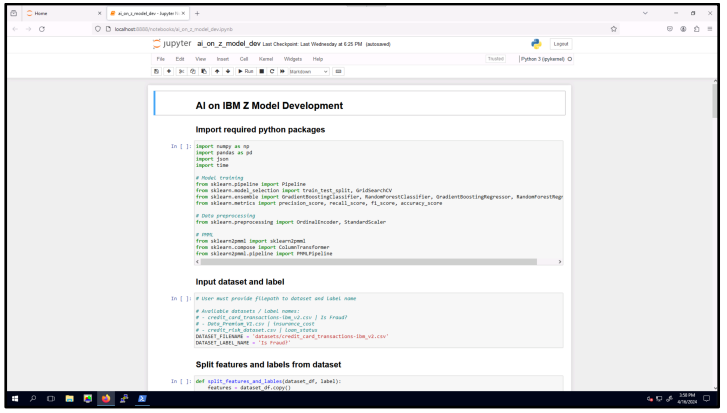
1. Your input dataset (csv) in `datasets/` directory
2. Add input data to Jupyter notebook (ai_on_z_model_dev.ipynb)
    a. Set `DATASET_FILENAME` to the path to your dataset
    b. Set `DATASET_LABEL_NAME` to the name of the column you're predicting from the dataset



Model training

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

    Note: This may take several minutes



Access rapid AI on Linux on Z development environment

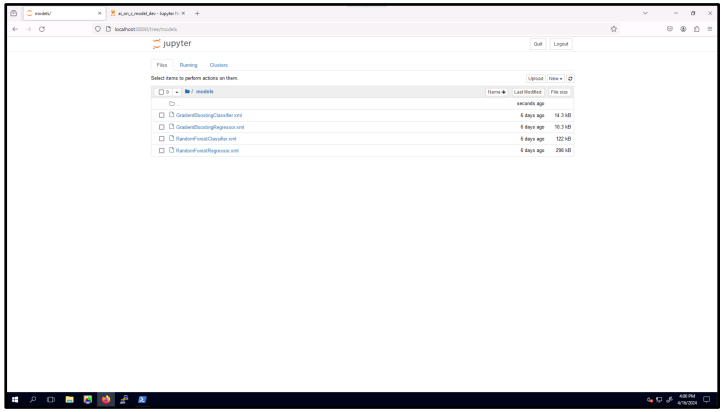
Provide data

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

Prerequisites

- Must have Docker or Podman installed

Step 2

AI model deployment

We will deploy our fraud detection AI model using TIS. We can utilize the AI Toolkit to leverage TIS for model deployment. This deployed AI model can then be integrated into applications within the Linux on Z environment.

Build Triton Inference Server

Integrate AI model into Triton Inference Server

Deploy Triton Inference Server

Run sample test

Build Triton Inference Server

1. Build podman image

   ```
   podman build -t zst-tis .
   ```

Integrate AI model into Triton Inference Server

1. Add your model (.pmml file) to

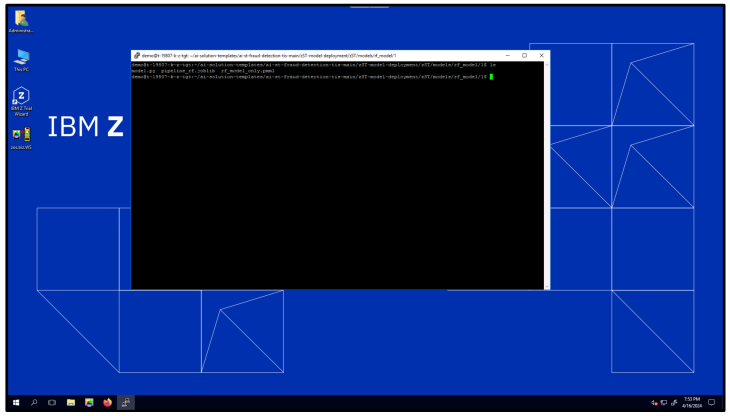
   `aionz-st-data-preprocessing-tis-main/zST-model-deployment/zST/models/rf_model/1`

   directory

2. Add your preprocessing .joblib file to

   `aionz-st-data-preprocessing-tis-main/zST-model-deployment/zST/models/rf_model/1`

   directory



Build Triton Inference Server

Integrate AI model into Triton Inference Server

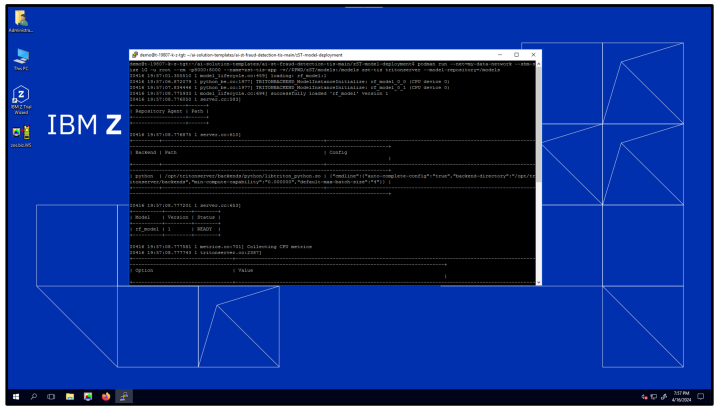
Deploy Triton Inference Server

Run sample test

Deploy Triton Inference Server

1. Run podman container

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

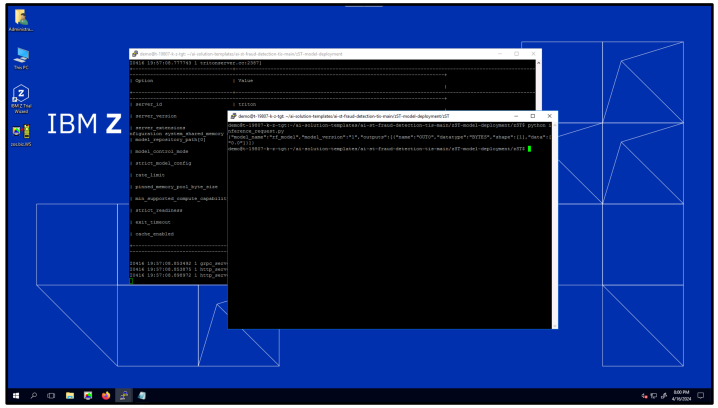


Run sample test

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

```
cd aionz-st-data-preprocessing-tis/zST-model-deployment/zST
```

```
python inference_request.py
```



☑ AI model deployment complete

Prerequisites

- Must have Docker or Podman installed

Step 3

Preprocessing deployment

We will deploy a preprocessing service using a Golang container. We can use this preprocessing service at inference time and send the data to Triton Inference Server.

All sample code for this section is within

`aionz-st-data-preprocessing-tis/zST-preprocess-deployment`

Access preprocessing service
Configure preprocessing service
Build preprocessing service
Deploy preprocessing service

Access preprocessing service

1. Access sample code

   ```
   cd zST-preprocess-deployment
   ```

Configure preprocessing service

1. Set the environment variables within

   ```
   aionz-st-data-preprocessing-tis/zST-preprocess-deployment/env.list
   ```

   SCORING_URL (scoring URL for deployed AI model)
   SCORING_PORT (scoring port for deployed AI model)

Build preprocessing service

1. Run command in terminal

   ```
   podman build -t go-preprocess .
   ```

Access preprocessing service
Configure preprocessing service
Build preprocessing service
Deploy preprocessing service

Deploy preprocessing service

1. Run command in terminal

```
podman run -p 8080:8080 --network="host"
--env-file env.list -t go-preprocess
```

☑ Preprocessing service deployment complete

Step 4

AI model inferencing

We can perform inferencing with our AI model that is deployed within Triton Inference Server. We will preprocess our data and then send it to Triton Inference Server to utilize our AI model.

All sample code for this section is within

`aionz-st-data-preprocessing-tis/zST-model-inferencing`

Prerequisites

- Must have TIS deployed
- Must have preprocessing service deployed
- Must have Python installed

Run inferencing

Run inferencing

1. Run sample application

```
python zst_preprocess_inferencing.py
```

☑ AI model inferencing complete