Model Usage Instructions

Group 10: Zhekai Zhang, Panfeng Jiang, Junyu Zhang, Jose Daniel Cantu Cantu

March 1, 2025

1 Introduction

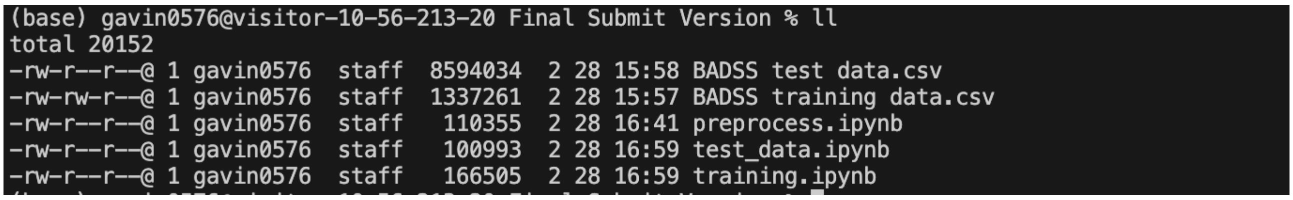
This document provides a detailed guide on how to use our machine learning model, including data preprocessing, training, and testing steps.

Things marked red are what you must do to correctly run our model.

Things marked blue are what you might want to do if you are to change some parameters in our model.

2 Files Overview

Our project consists of several key files that are crucial for data handling, model training, and testing:



```
(base) gavin0576@visitor-10-56-213-20 Final Submit Version % ll
total 20152
-rw-r--r--@ 1 gavin0576  staff  8594034  2 28 15:58 BADSS test data.csv
-rw-rw-r--@ 1 gavin0576  staff  1337261  2 28 15:57 BADSS training data.csv
-rw-r--r--@ 1 gavin0576  staff   110355  2 28 16:41 preprocess.ipynb
-rw-r--r--@ 1 gavin0576  staff   100993  2 28 16:59 test_data.ipynb
-rw-r--r--@ 1 gavin0576  staff   166505  2 28 16:59 training.ipynb
```

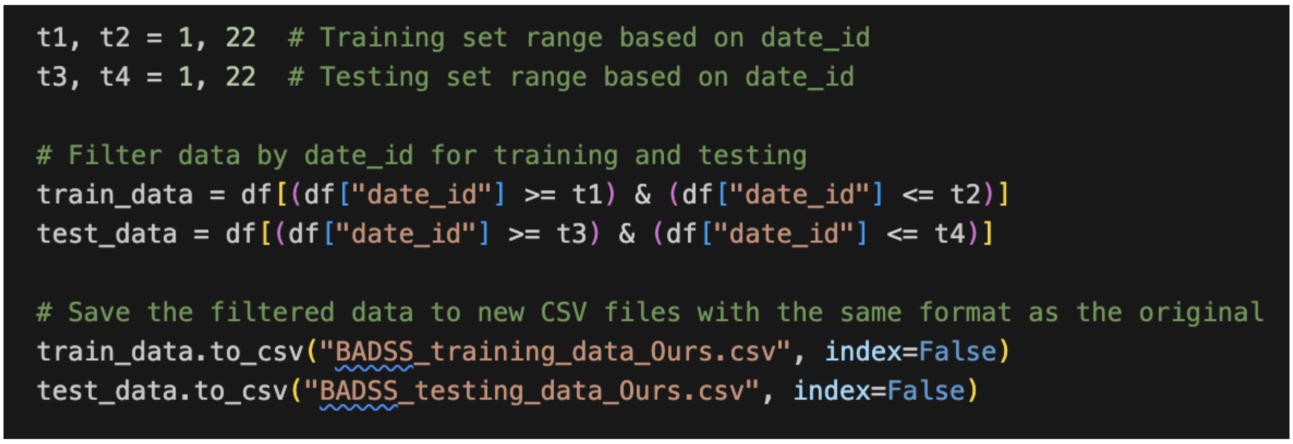
Figure 1: files in the zip

- **BADSS test data.csv**: This is the dataset used for training and testing our model. It contains multiple features and labels required for model learning.
- **BADSS data training.csv**: This is the dataset used for training and testing our model. It contains multiple features and labels required for model learning.
- **preprocess.ipynb**: This Jupyter Notebook handles data preprocessing, including cleaning, feature extraction, and transformation.
- **training.ipynb**: This Notebook is responsible for model training. It loads the processed data, trains the model, and saves the trained weights.
- **test_data.ipynb**: This Notebook is used for model evaluation and testing. It applies the trained model to test data and outputs performance metrics.

3 Data Preprocessing(First Step)

To preprocess the data, simply open **preprocess.ipynb** and run all cells. This script will perform the following operations:

- Fill in missing values to ensure data completeness.
- Conduct data screening to filter out inconsistent or irrelevant data.
- Discretize date fields into integer values for model compatibility.
- Compute **PnL** and **Exposure** as part of feature engineering.
- If required, split the dataset into training and testing sets:
  - **BADSS_training_data_Ours.csv**: The training dataset after removing specified test days.
  - **BADSS_testing_data_Ours.csv**: The testing dataset containing only specified test days.
- If you want to train on data from **day_id1 to day_id2**, modify `t1` and `t2` at the end of the script.
- If you want to test on data from **day_id3 to day_id4**, modify `t3` and `t4` at the end of the script.



```
t1, t2 = 1, 22  # Training set range based on date_id
t3, t4 = 1, 22  # Testing set range based on date_id

# Filter data by date_id for training and testing
train_data = df[(df["date_id"] >= t1) & (df["date_id"] <= t2)]
test_data = df[(df["date_id"] >= t3) & (df["date_id"] <= t4)]

# Save the filtered data to new CSV files with the same format as the original
train_data.to_csv("BADSS_training_data_Ours.csv", index=False)
test_data.to_csv("BADSS_testing_data_Ours.csv", index=False)
```

Figure 2: Configuration for Training and Testing Date Range

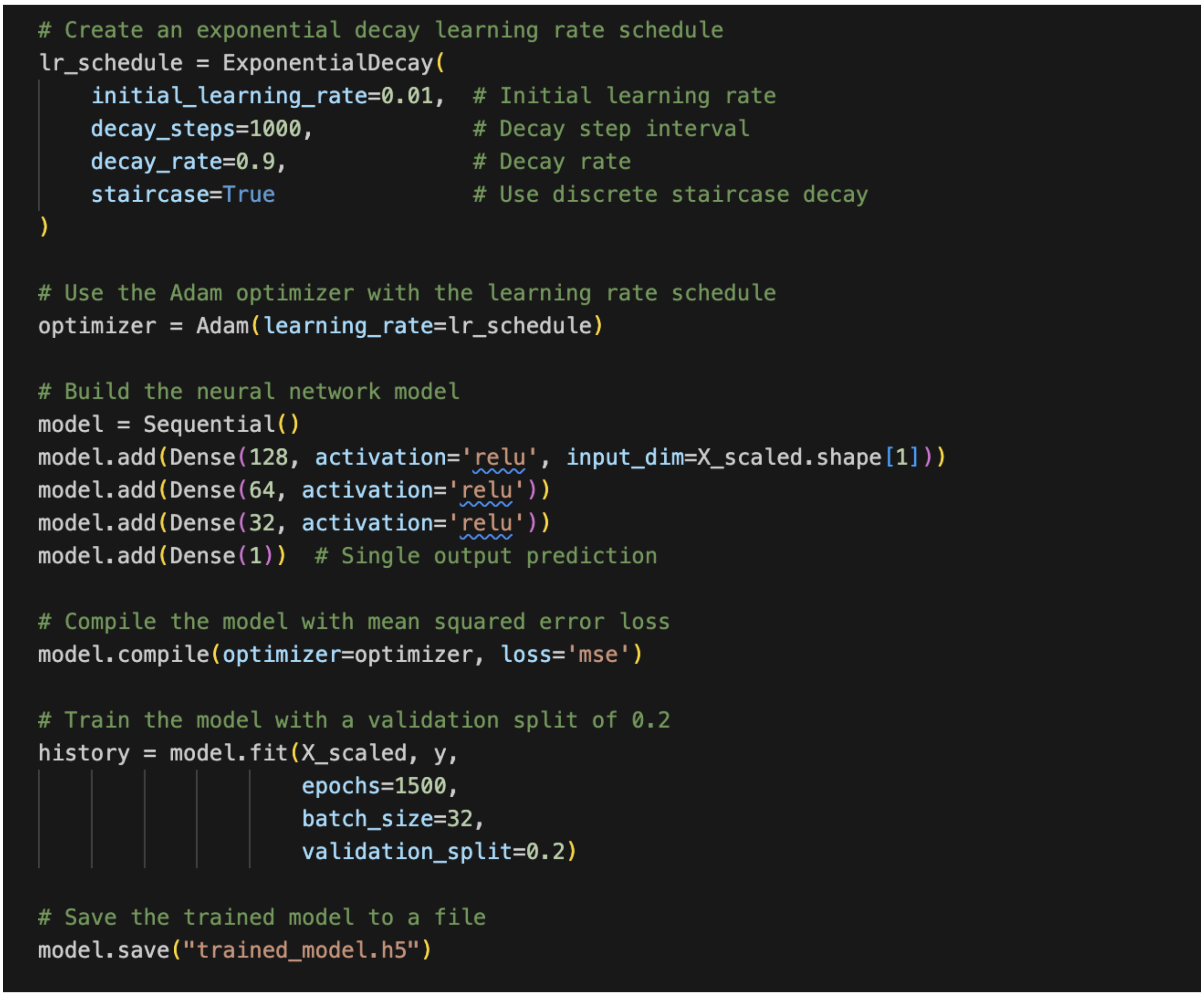
After executing the preprocessing script, the processed data will be ready for model training.

4 Model Training(Second Step)

To train the model, simply open **training.ipynb** and click the **run all**. This script will:

- Load the preprocessed data from **BADSS_training_data_Ours.csv**.
- Train the model using the dataset, optimizing the network parameters.
- Save the trained model in **trained_model.h5**, which contains all necessary parameters for inference.

- Training typically takes **5-10 minutes**, depending on computational resources.
- Once training is complete, the left panel will display the **trained_model.h5** file.
- This model file will be used for evaluating test data in later steps.
- If you want to change the parameter of our training model, like the structure of our multi-layer perceptron, learning rate, batch size, epochs or any other things you want to modify, simply see the instructions written in the code here, and modify the number belongs to each parameter in your needs.



```python
# Create an exponential decay learning rate schedule
lr_schedule = ExponentialDecay(
    initial_learning_rate=0.01,  # Initial learning rate
    decay_steps=1000,            # Decay step interval
    decay_rate=0.9,              # Decay rate
    staircase=True               # Use discrete staircase decay
)

# Use the Adam optimizer with the learning rate schedule
optimizer = Adam(learning_rate=lr_schedule)

# Build the neural network model
model = Sequential()
model.add(Dense(128, activation='relu', input_dim=X_scaled.shape[1]))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(1))  # Single output prediction

# Compile the model with mean squared error loss
model.compile(optimizer=optimizer, loss='mse')

# Train the model with a validation split of 0.2
history = model.fit(X_scaled, y,
                    epochs=1500,
                    batch_size=32,
                    validation_split=0.2)

# Save the trained model to a file
model.save("trained_model.h5")
```

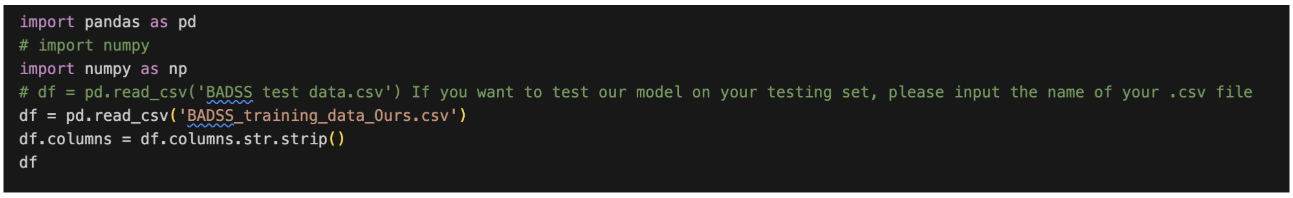
Figure 3: Configuration for Our Model

5 Model Testing(Third Step)

To evaluate the trained model, open **test_data.ipynb** and run all cells. This script will test the model under three different scenarios:

- **Overfitting Test**: The model is tested in an overfitting scenario to determine the best possible performance. The results are displayed within the Jupyter Notebook.

- **Baseline Test**: The model randomly selects options to purchase and calculates the associated costs and exposure, providing a comparison point for evaluation.
- **Trained Model Performance(This is exactly what we want)**: The trained model is applied to the test data, and its performance is evaluated by calculating the costs and exposure. The final purchase strategy results are saved in a **.txt** file.
- If you want to test the model in the datasets generated from part of the training set, just leave the file name as "BADSS_training_data_Ours.csv". Otherwise, if you want to test your private testing data, just change the file name and then click run all to run on your personal dataset.



```
import pandas as pd
# import numpy
import numpy as np
# df = pd.read_csv('BADSS test data.csv') If you want to test our model on your testing set, please input the name of your .csv file
df = pd.read_csv('BADSS_training_data_Ours.csv')
df.columns = df.columns.str.strip()
df
```

Figure 4: You might want to change the file name to run your own datasets

The results from the testing phase will help assess the effectiveness of the trained model in comparison to random baseline strategies.