Week 2 Attempt Summary

Introduction

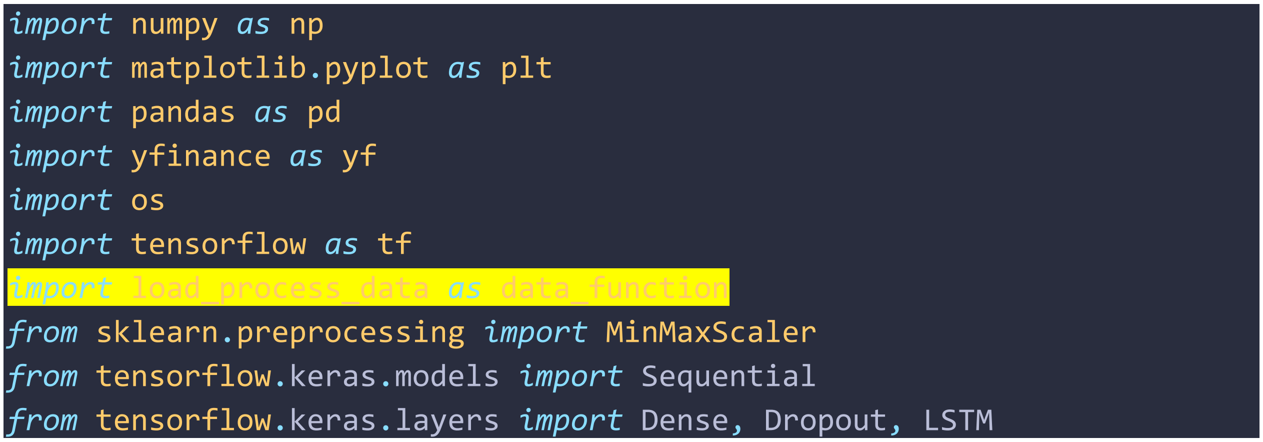
In this week we were advised to create a load and process data function to a dataset with multiple features, with the following functional requirements

- Should be able to specify and start and end date for the whole dataset
- Deal with Nan values in the dataset
- Split train/test data according to a specified ratio (Date)
- Should be able to download the data to the local machine and store them to future use
- Scaling in the data structure

Load_and_process_data Function

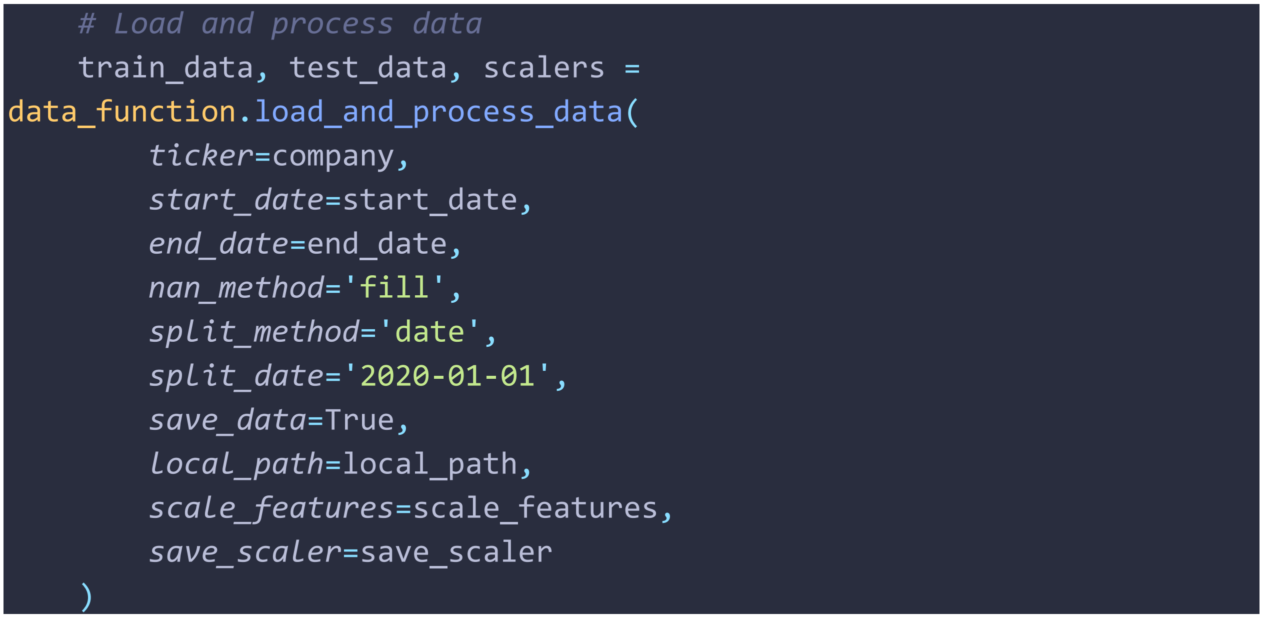
```
import pandas as pd
import yfinance as yf
import os
from sklearn.preprocessing import MinMaxScaler

def load_and_process_data(ticker, start_date, end_date,
nan_method='drop', split_method='ratio',
                          train_size=0.8, test_size=0.2,
split_date=None, random_state=None,
                          save_data=False, local_path=None,
scale_features=False, save_scaler=False):

    # Load data locally if available
    if local_path and os.path.exists(local_path):
        data = pd.read_csv(local_path, index_col='Date',
parse_dates=True)
    else:
        # Download the data
        data = yf.download(ticker, start=start_date, end=end_date)
        # Save the data locally
        if save_data and local_path:
            data.to_csv(local_path)

    # Handle NaN values
    if nan_method == 'drop':
        data = data.dropna()
    elif nan_method == 'fill':
        data = data.ffill().bfill()

    # Ensure data is not empty
    if data.empty:
        raise ValueError(f"No data found for {ticker} between
{start_date} and {end_date}. Please check the date range and ticker
symbol.")

    # Split the data
    if split_method == 'ratio':
        train_data, test_data = train_test_split(data,
train_size=train_size, test_size=test_size, random_state=random_state,
shuffle=True)
    elif split_method == 'date' and split_date:
        train_data = data[:split_date]
        test_data = data[split_date:]

    # Scale features if required
    scalers = {}
    if scale_features:
        for column in train_data.columns:
            scaler = MinMaxScaler(feature_range=(0, 1))
            train_data[column] =
scaler.fit_transform(train_data[[column]])
            if not test_data.empty:
                test_data[column] =
scaler.transform(test_data[[column]])
            if save_scaler:
                scalers[column] = scaler

    return train_data, test_data, scalers
```

This function is in a different python file (load_process_data.py). I am importing the function with a import statement at the top



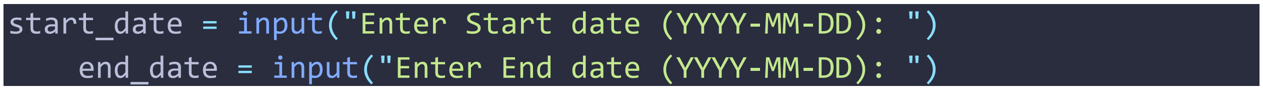
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import yfinance as yf
import os
import tensorflow as tf
import load_process_data as data_function
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM
```

And this to instantiate it in the main python script



```
    # Load and process data
    train_data, test_data, scalers =
data_function.load_and_process_data(
        ticker=company,
        start_date=start_date,
        end_date=end_date,
        nan_method='fill',
        split_method='date',
        split_date='2020-01-01',
        save_data=True,
        local_path=local_path,
        scale_features=scale_features,
        save_scaler=save_scaler
    )
```

This week 2 code base allows the user to enter the start date and the end date manually through the terminal before training the model



```
start_date = input("Enter Start date (YYYY-MM-DD): ")
    end_date = input("Enter End date (YYYY-MM-DD): ")
```

And It handles the Nan values with predefined Nan value handlers that comes with the pandas library

Dropna :- Removes data Frames that contains Nan values

Fillna :- Fills missing data Frames with a specified value

And with with some exception handling we have this code

```
# Handle NaN values
    if nan_method == 'drop':
        data = data.dropna()
    elif nan_method == 'fill':
        data = data.ffill().bfill()

    # Ensure data is not empty
    if data.empty:
        raise ValueError(f"No data found for {ticker} between
{start_date} and {end_date}. Please check the date range and ticker
symbol.")
```

it raises a Error massage when ever the data array might be empty (for debugging pruposes)

And this code facilitates the functionality to split the data into 2, Training and testing data

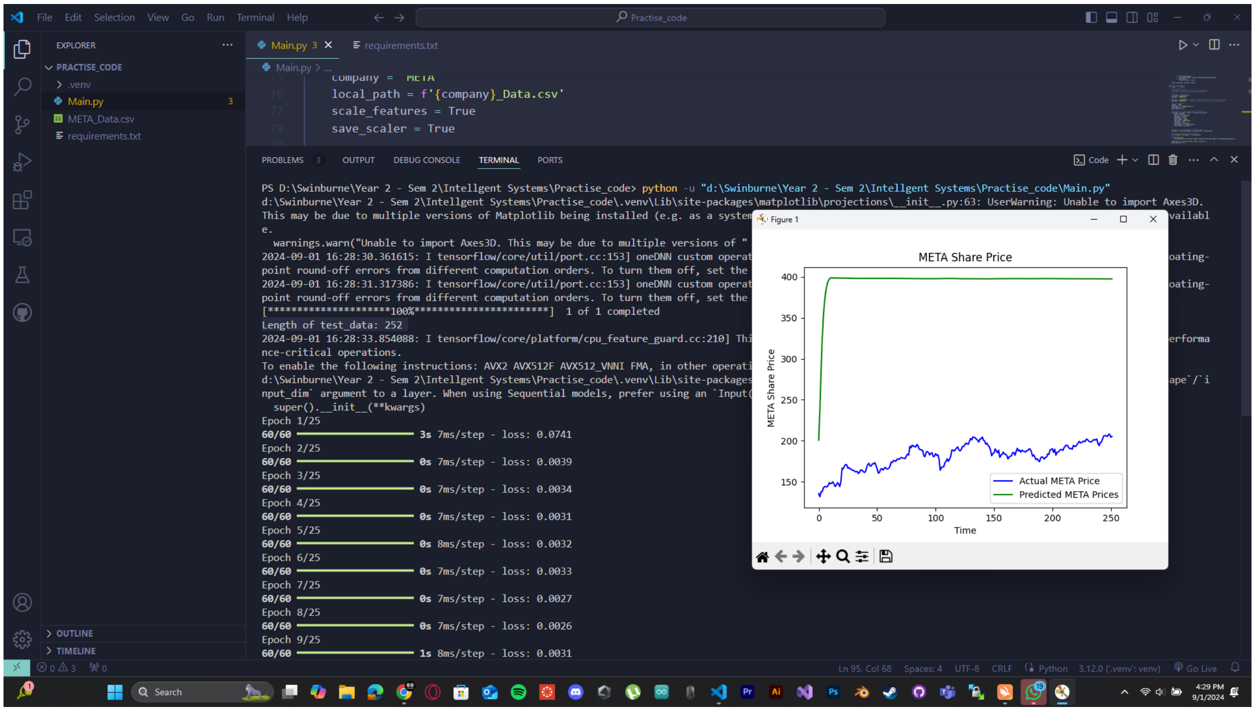
```
# Split the data
    if split_method == 'ratio':
        train_data, test_data = train_test_split(data,
train_size=train_size, test_size=test_size, random_state=random_state,
shuffle=True)
    elif split_method == 'date' and split_date:
        train_data = data[:split_date]
        test_data = data[split_date:]
```

Yet the train_test_split function is still under development, for this week the date method will be used.

And then the last requirement was for adding scalers to the code base

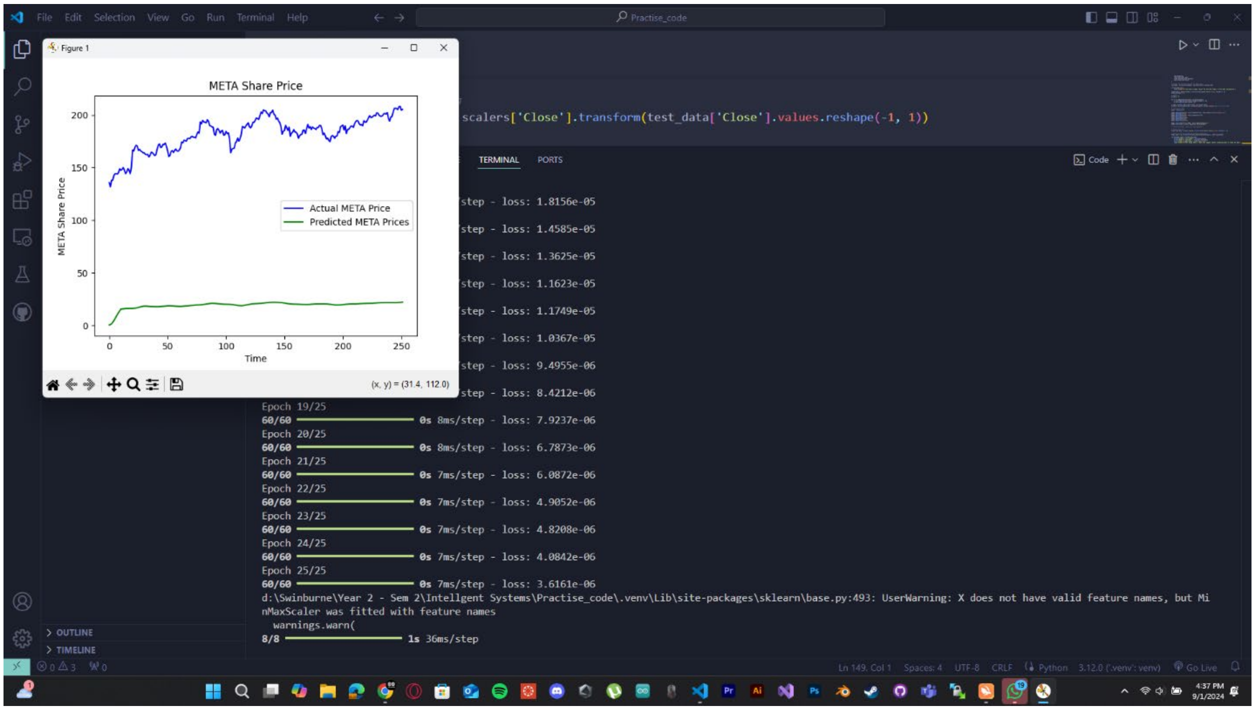
```
# Scale features if required
    scalers = {}
    if scale_features:
        for column in train_data.columns:
            scaler = MinMaxScaler(feature_range=(0, 1))
            train_data[column] =
scaler.fit_transform(train_data[[column]])
            if not test_data.empty:
                test_data[column] =
scaler.transform(test_data[[column]])
            if save_scaler:
                scalers[column] = scaler
```

I had problems with using the "Adam" optimizer when compiling the model



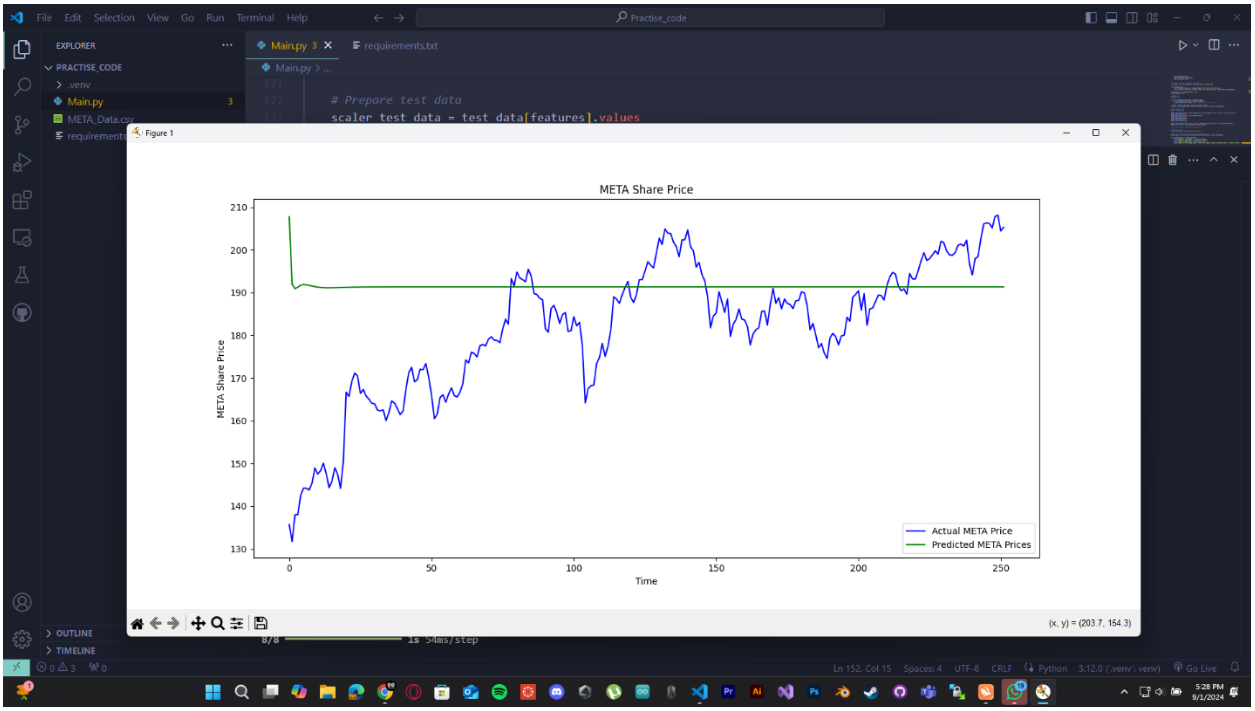
```
    company = 'META'
    local_path = f'{company}_Data.csv'
    scale_features = True
    save_scaler = True
```

```
PS D:\Swinburne\Year 2 - Sem 2\Intellgent Systems\Practise_code> python -u "d:\Swinburne\Year 2 - Sem 2\Intellgent Systems\Practise_code\Main.py"
d:\Swinburne\Year 2 - Sem 2\Intellgent Systems\Practise_code\.venv\Lib\site-packages\matplotlib\projections\__init__.py:63: UserWarning: Unable to import Axes3D.
This may be due to multiple versions of Matplotlib being installed (e.g. as a system                                  vailabl
e.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions of "
2024-09-01 16:28:30.361615: I tensorflow/core/util/port.cc:153] oneDNN custom operat                                  oating-
point round-off errors from different computation orders. To turn them off, set the
2024-09-01 16:28:31.317386: I tensorflow/core/util/port.cc:153] oneDNN custom operat                                  oating-
point round-off errors from different computation orders. To turn them off, set the
[*********************100%***********************]  1 of 1 completed
Length of test_data: 252
2024-09-01 16:28:33.854088: I tensorflow/core/platform/cpu_feature_guard.cc:210] Thi                                  erforma
nce-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operati
d:\Swinburne\Year 2 - Sem 2\Intellgent Systems\Practise_code\.venv\Lib\site-packages                                  ape`/`i
nput_dim` argument to a layer. When using Sequential models, prefer using an `Input(
  super().__init__(**kwargs)
Epoch 1/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 0.0741
Epoch 2/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0039
Epoch 3/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0034
Epoch 4/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0031
Epoch 5/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 0.0032
Epoch 6/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0033
Epoch 7/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0027
Epoch 8/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 0.0026
Epoch 9/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - loss: 0.0031
```



```
scalers['Close'].transform(test_data['Close'].values.reshape(-1, 1))
```

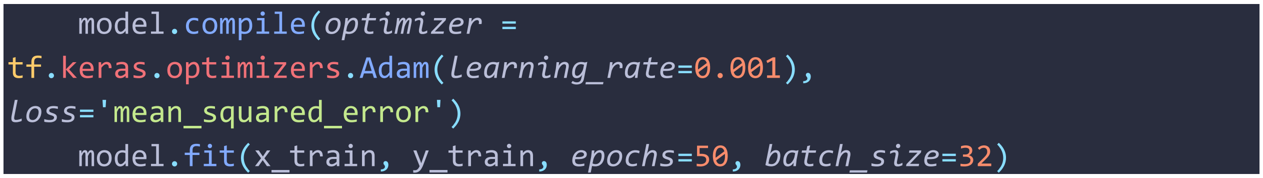
```
step - loss: 1.8156e-05
step - loss: 1.4585e-05
step - loss: 1.3625e-05
step - loss: 1.1623e-05
step - loss: 1.1749e-05
step - loss: 1.0367e-05
step - loss: 9.4955e-06
step - loss: 8.4212e-06
Epoch 19/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 7.9237e-06
Epoch 20/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - loss: 6.7873e-06
Epoch 21/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 6.0872e-06
Epoch 22/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.9052e-06
Epoch 23/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.8208e-06
Epoch 24/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 4.0842e-06
Epoch 25/25
60/60 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - loss: 3.6161e-06
d:\Swinburne\Year 2 - Sem 2\Intellgent Systems\Practise_code\.venv\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but Mi
nMaxScaler was fitted with feature names
  warnings.warn(
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step
```



```
# Prepare test data
scaler_test_data = test_data[features].values
```

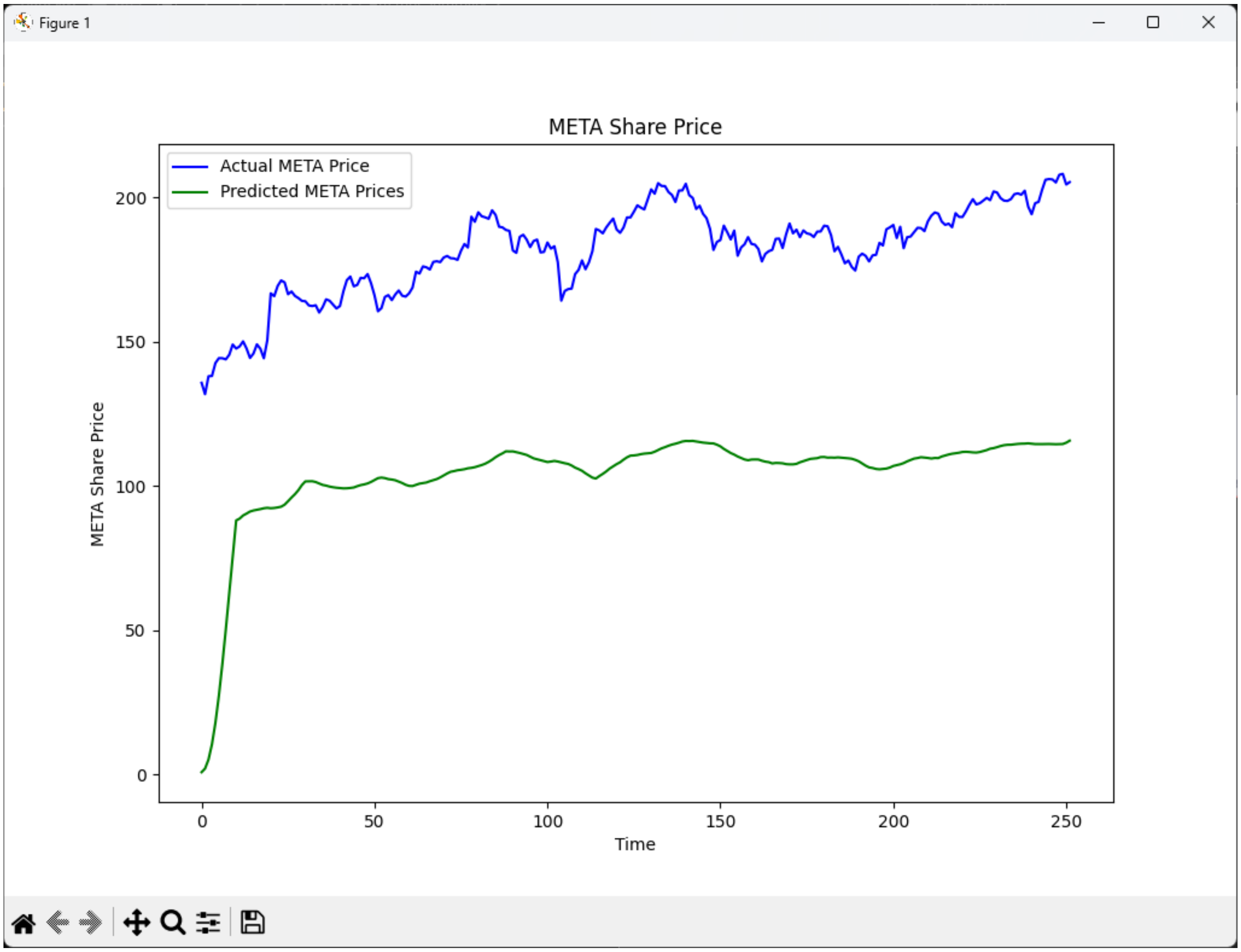
```
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 54ms/step
```

Then I changed the ‘Adam’ optimizer and used tensor flow keras adam optimizer instead, with 100 epochs



```
    model.compile(optimizer =
tf.keras.optimizers.Adam(learning_rate=0.001),
loss='mean_squared_error')
    model.fit(x_train, y_train, epochs=50, batch_size=32)
```

By using that I was able to take a reasonable output



But this is very far from being perfect but this is still a work in progress