OPTION B - SET UP

TRAN DUC ANH DANG

Student ID: 103995439

Summaries of Environment Setup

V0.1 (stock-prediction.py) provided by the school seems like it has been saved as python file rather than Jupyter Notebook, therefore I have put the code in Jupyter Notebook and also have some code modified for a better usage later on in this project. In both V0.1 and P1 both use commands like !pip install to install necessary Python packages and libraries. This is common in a Colab environment, which is a cloud-based Jupyter Notebook interface provided by Google or local Jupyter Notebook interface using Anaconda.

Requirements

Virtual Environment

- Google Colab
- Jupyter Notebook

Dependencies (v0.1)

- numpy
- matplotlib
- mplfinance
- pandas
- scikit-learn
- pandas-datareader
- yfinance

Dependencies (P1)

- numpy
- matplotlib
- mplfinance
- pandas
- scikit-learn
- yahoo_fin

Dependencies (P2)

- numpy
- mplfinance
- pandas
- pandas-datareader
- yfinance
- matplotlib
- arrow

Installation

*Note: Anaconda is required unless Google Collab is being used

Anaconda/Virtual Environment

1. Download Anaconda: Go to the Anaconda website (https://www.anaconda.com/products/distribution) and download the appropriate version for your operating system.
2. Install Anaconda: Follow the installation instructions for the OS from the Anaconda website.

3. Open Anaconda Navigator: Launch Anaconda Navigator from your installed applications.
4. Create a New Environment (Required): Create a new environment to isolate Jupyter installation on each project. Click on "Environments" in Navigator and then "Create" to make a new environment.
5. Install Jupyter Notebook: In the selected environment, click on the environment name and select "Open Terminal". In the terminal, type: conda install jupyter.

Dependencies

In Google Colab or Jupyter Notebook, it can directly install the required dependencies using the !pip command in code cells. Here's an example of how to install the dependencies:

```
!pip install <package> or !pip install -r <text file>
```

Understanding of the Initial Codebase

In P1, the codebase seems to be focused on predicting stock prices using deep learning. It likely uses TensorFlow to define, train, and evaluate a neural network model. The initial steps involve data loading and pre-processing, establishing the foundation for the subsequent modelling stages. The emphasis seems to be on a singular company stock prediction, with specific configurations and setups geared towards that.

In V0.1, the codebase also revolves around stock price prediction. However, there might be subtle differences in the model architecture, data handling, or the prediction approach compared to p1.py. The essence remains the same: use historical stock price data to train a model and predict future prices. The various libraries included, such as pandas-datareader and yfinance, indicate a comprehensive approach to data extraction and manipulation.

Modified V0.1

In V0.1, I have modified a few things but keep the same content such as:

1. Visualisation
   - The modified version introduces superior visualization techniques, enhancing clarity and interpretability of the stock prediction results.
   - A dedicated plotting function provides an intuitive representation of actual vs. predicted prices, aiding in a quick visual assessment of the model's performance.

   ```
   plt.plot(actual_prices, color="black", label=f"Actual {COMPANY} Price")
   plt.plot(predicted_prices, color="green", label=f"Predicted {COMPANY} Price")
   plt.title(f"{COMPANY} Share Price")
   plt.xlabel("Time")
   plt.ylabel(f"{COMPANY} Share Price")
   plt.legend()
   plt.show()
   ```

2. Predictions for Future Dates

   - A standout feature in the modified version is its ability to predict stock prices for specified future dates.
   - This forward-looking capability can be instrumental for traders and analysts aiming to strategize for the upcoming days.

   future_x = len(actual_prices)
   plt.scatter(future_x, prediction.flatten(), color='orange', label='Future Predictions')

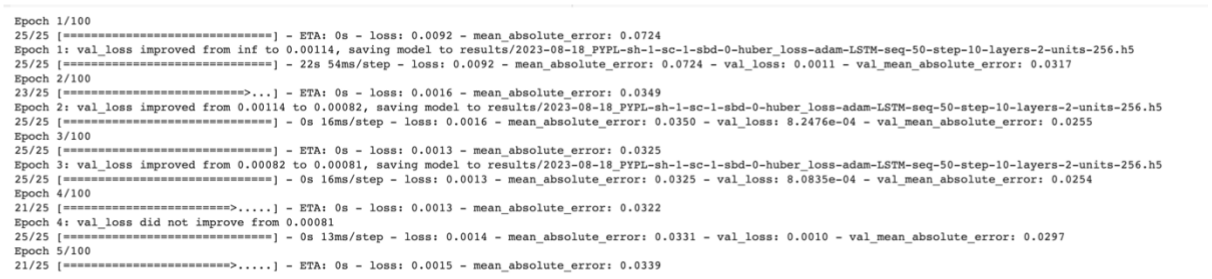
3. Iterative Predictions for Multiple Companies
   - The modified version underscores the essence of scalability in stock market predictions. Recognizing that companies like NVDA, TSLA, and AAPL often exhibit movements influenced by major indices such as NASDAQ, it is equipped to handle predictions for these companies iteratively. This iterative approach not only offers versatility but also accounts for potential correlated movements between companies and indices, making predictions more insightful and aligned with real-world market dynamics.

   companies = ["NVDA", "TSLA", "AAPL"]
   for COMPANY in companies:
   **   latest_data = yf.download(COMPANY, start=TEST_START, end=TEST_END)**

P1

The P1 codebase is a meticulously crafted script, designed to traverse the intricate journey of stock market predictions. From raw data extraction to the nuanced intricacies of deep learning, the code offers a spectrum of functionalities. But the true essence of its capabilities is best visualised and captured in its outputs.

Training Process:



```
Epoch 1/100
25/25 [==============================] - ETA: 0s - loss: 0.0092 - mean_absolute_error: 0.0724
Epoch 1: val_loss improved from inf to 0.00114, saving model to results/2023-08-18_PYPL-sh-1-sc-1-sbd-0-huber_loss-adam-LSTM-seq-50-step-10-layers-2-units-256.h5
25/25 [==============================] - 22s 54ms/step - loss: 0.0092 - mean_absolute_error: 0.0724 - val_loss: 0.0011 - val_mean_absolute_error: 0.0317
Epoch 2/100
23/25 [===========================>...] - ETA: 0s - loss: 0.0016 - mean_absolute_error: 0.0349
Epoch 2: val_loss improved from 0.00114 to 0.00082, saving model to results/2023-08-18_PYPL-sh-1-sc-1-sbd-0-huber_loss-adam-LSTM-seq-50-step-10-layers-2-units-256.h5
25/25 [==============================] - 0s 16ms/step - loss: 0.0016 - mean_absolute_error: 0.0350 - val_loss: 8.2476e-04 - val_mean_absolute_error: 0.0255
Epoch 3/100
25/25 [==============================] - ETA: 0s - loss: 0.0013 - mean_absolute_error: 0.0325
Epoch 3: val_loss improved from 0.00082 to 0.00081, saving model to results/2023-08-18_PYPL-sh-1-sc-1-sbd-0-huber_loss-adam-LSTM-seq-50-step-10-layers-2-units-256.h5
25/25 [==============================] - 0s 16ms/step - loss: 0.0013 - mean_absolute_error: 0.0325 - val_loss: 8.0835e-04 - val_mean_absolute_error: 0.0254
Epoch 4/100
21/25 [=========================>.....] - ETA: 0s - loss: 0.0013 - mean_absolute_error: 0.0322
Epoch 4: val_loss did not improve from 0.00081
25/25 [==============================] - 0s 13ms/step - loss: 0.0014 - mean_absolute_error: 0.0331 - val_loss: 0.0010 - val_mean_absolute_error: 0.0297
Epoch 5/100
21/25 [=========================>.....] - ETA: 0s - loss: 0.0015 - mean_absolute_error: 0.0339
```

Figure P1.1.1 - Training Process

Graphical Output:

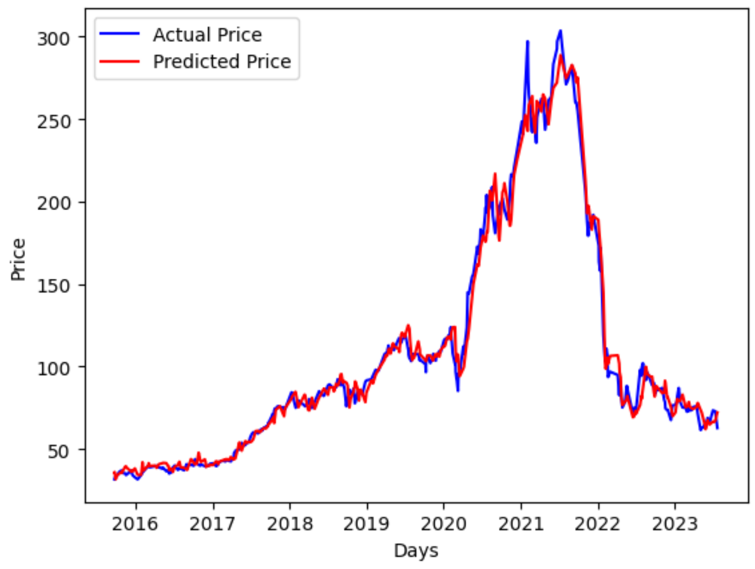
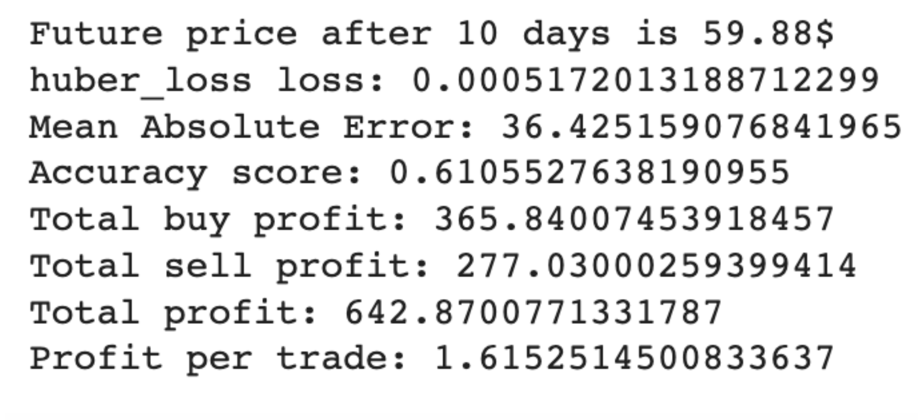


Figure P1.2.1 - True/Predict prices graph

- Historical Data Visualisation: A depiction of stock price movements over time, offering a foundational understanding of past stock behaviours.
- Model's Predictive Performance: Graphs contrasting the model's predictions against actual stock prices, providing an intuitive assessment of the model's accuracy.

Numerical Output:

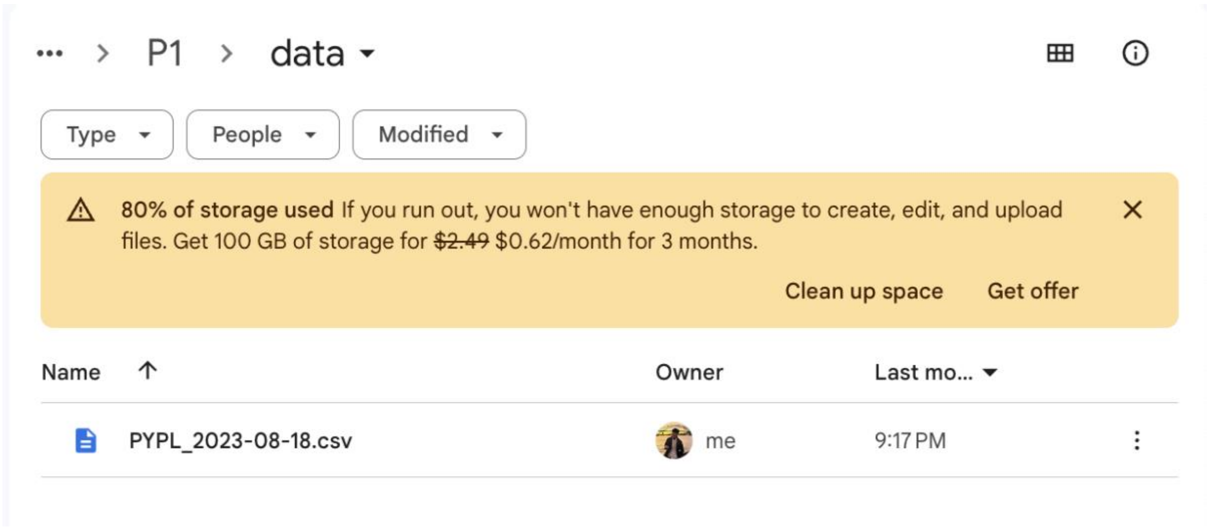


```
Future price after 10 days is 59.88$
huber_loss loss: 0.0005172013188712299
Mean Absolute Error: 36.425159076841965
Accuracy score: 0.6105527638190955
Total buy profit: 365.84007453918457
Total sell profit: 277.03000259399414
Total profit: 642.8700771331787
Profit per trade: 1.6152514500833637
```

Figure P1.3.1 - Numerical Output

- Model Evaluation Metrics: Be on the lookout for metrics such as Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE), offering quantitative insights into the model's performance.
- Predicted Stock Prices: Direct numerical predictions for future stock prices provide concrete values that can be used for investment strategies or further analysis.

Exported Files:


··· > P1 > data

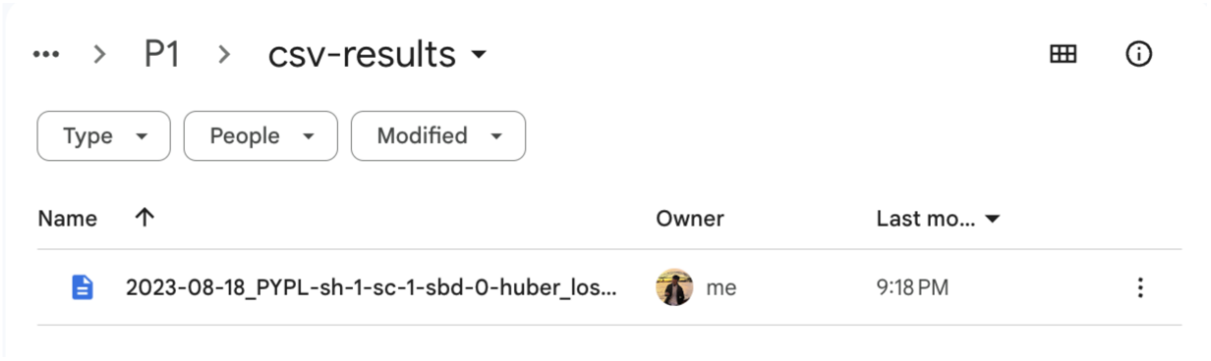
Type People Modified

80% of storage used If you run out, you won't have enough storage to create, edit, and upload files. Get 100 GB of storage for ~~$2.49~~ $0.62/month for 3 months.

Clean up space Get offer

Name	Owner	Last mo...
PYPL_2023-08-18.csv	me	9:17 PM

Figure P1.4.1 - Historical CSV File


··· > P1 > csv-results

Type People Modified

Name	Owner	Last mo...
2023-08-18_PYPL-sh-1-sc-1-sbd-0-huber_los...	me	9:18 PM

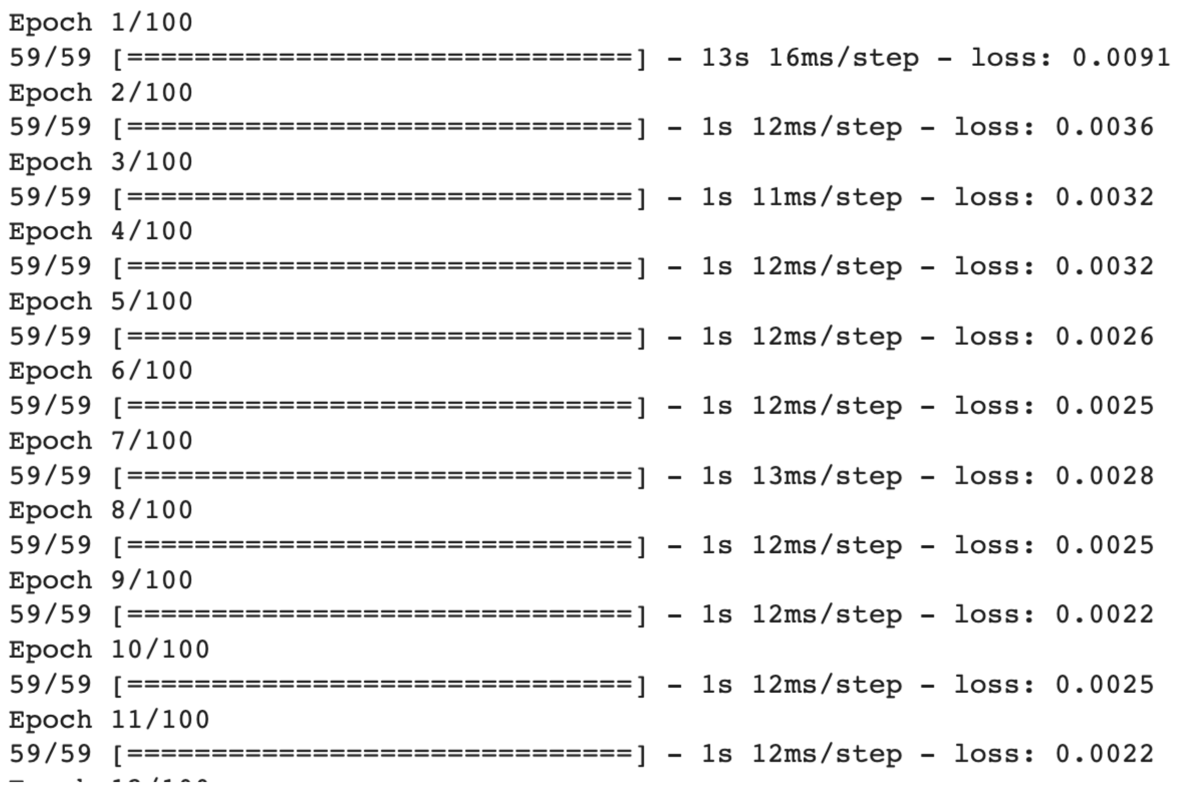
Figure P1.4.2 - Result CSV File

- CSV Exports: At various junctures, the code might export processed data or predictions in CSV format. These files serve as a permanent record, facilitating further analysis, sharing, or integration into other systems.

V0.1

In V0.1 script is a testament to the continuous evolution in the realm of stock market predictions. Building on its predecessor, this modified version introduces a slew of enhancements, each designed to offer a richer, more insightful analytical experience. The true measure of its advancements, however, is best gauged through its outputs.

Training Process:



```
Epoch 1/100
59/59 [==============================] - 13s 16ms/step - loss: 0.0091
Epoch 2/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0036
Epoch 3/100
59/59 [==============================] - 1s 11ms/step - loss: 0.0032
Epoch 4/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0032
Epoch 5/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0026
Epoch 6/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0025
Epoch 7/100
59/59 [==============================] - 1s 13ms/step - loss: 0.0028
Epoch 8/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0025
Epoch 9/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0022
Epoch 10/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0025
Epoch 11/100
59/59 [==============================] - 1s 12ms/step - loss: 0.0022
```

Figure P1.5.1 - Training Process

Graphical Output:

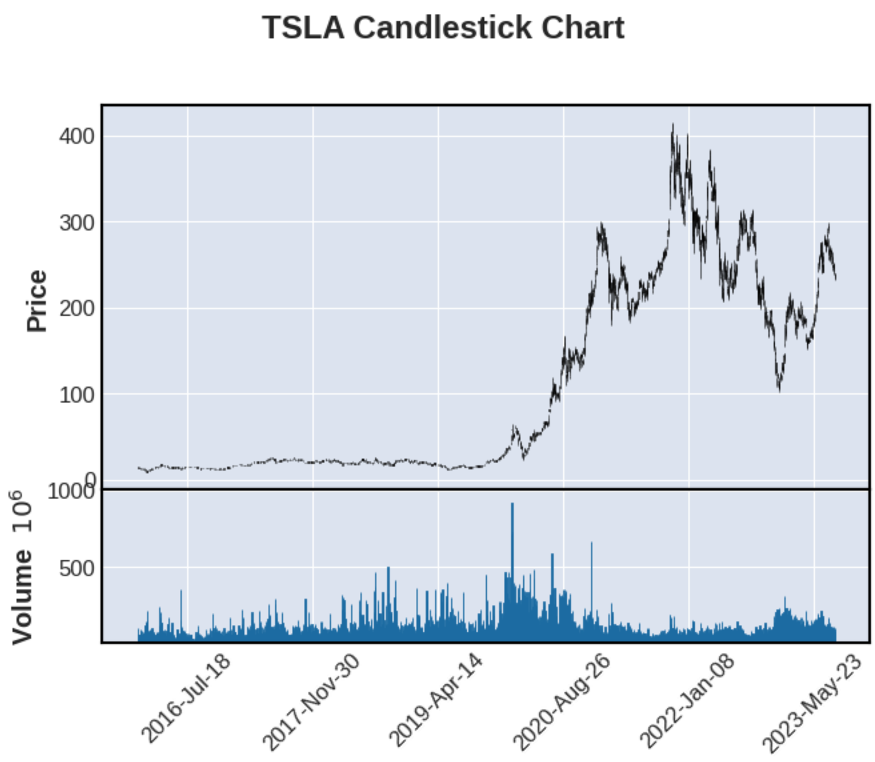


Figure P1.6.1 - Candlestick graph

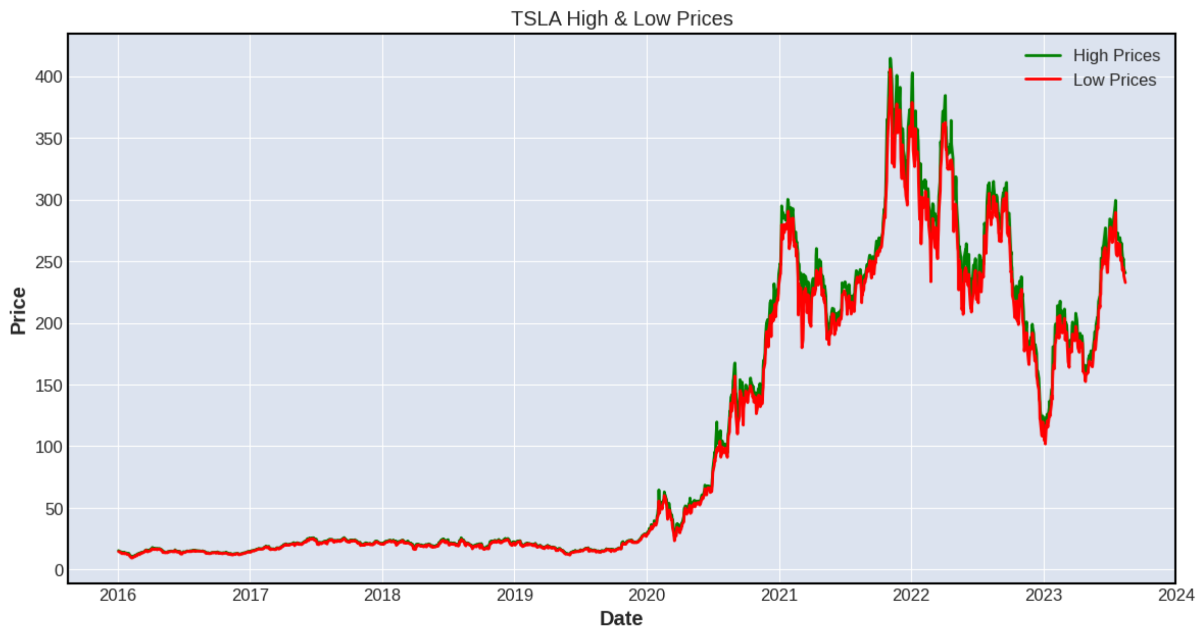


Figure P1.6.2 - High & Low Prices

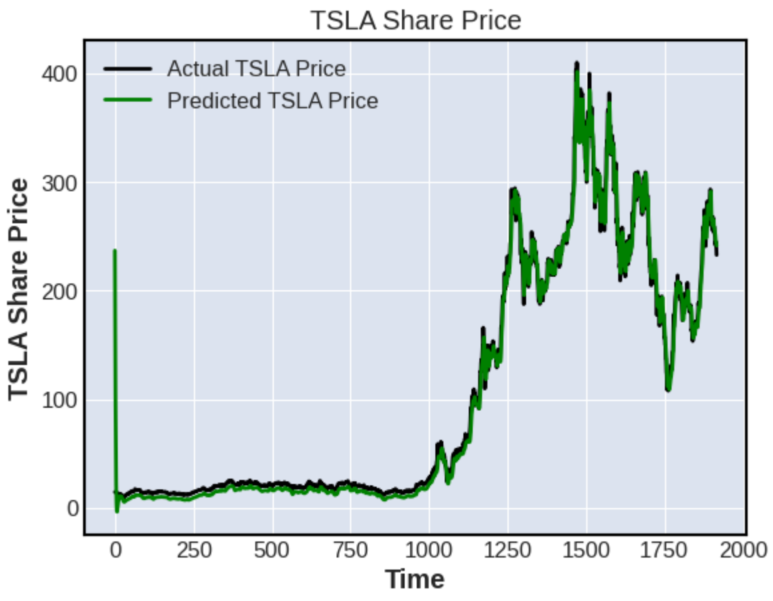


Figure P1.6.3 - True/Predict prices graph

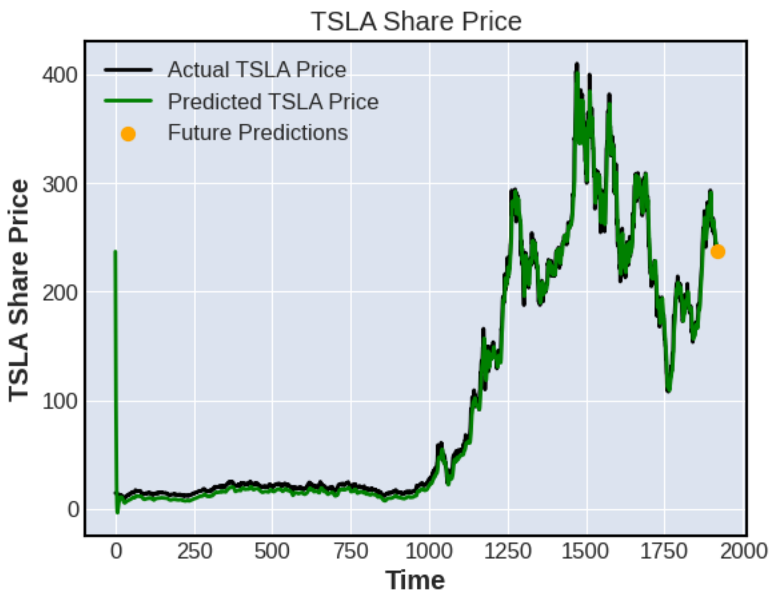


Figure P1.6.3 - True/Predict/Future prices graph

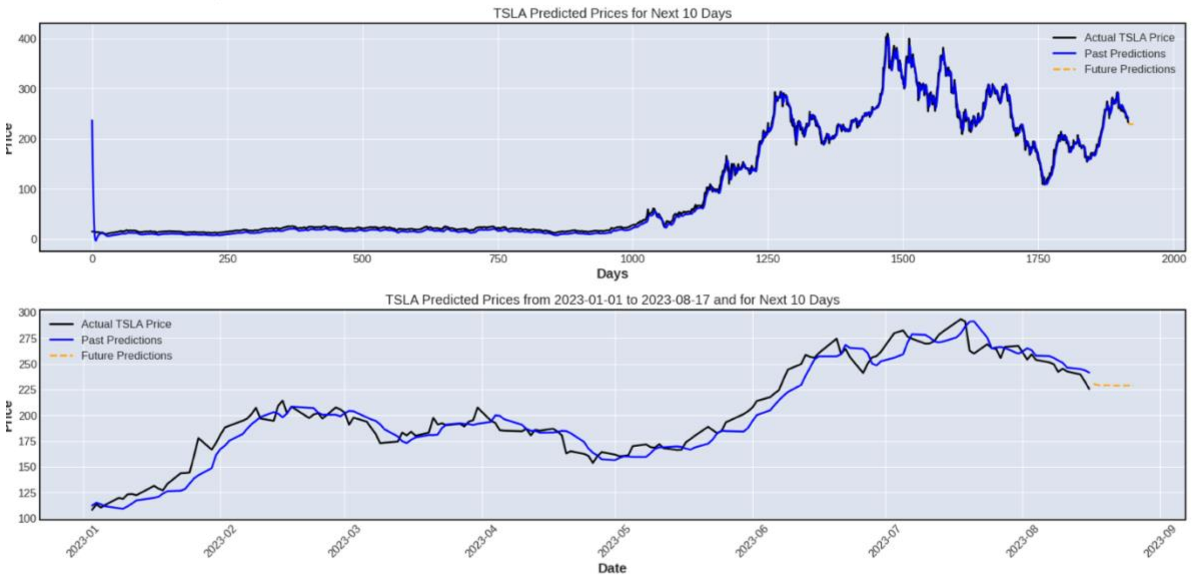
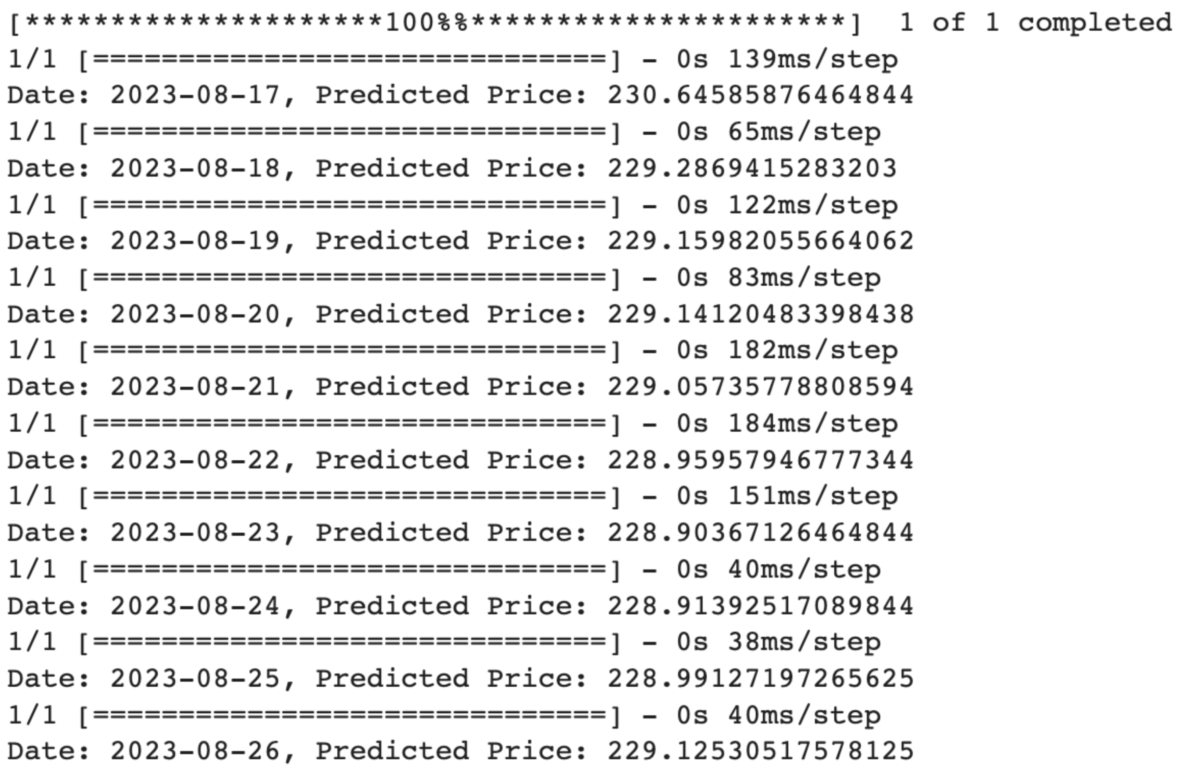


Figure P1.6.4 - 10 upcoming days prediction

- Enhanced Historical Data Visualization: While the historical data remains a cornerstone, the visualization techniques are refined, offering a clearer, more detailed view of stock price trends over time.
- Predictive Performance: The juxtaposition of predicted versus actual stock prices takes a front seat, with enhanced visual techniques ensuring a sharper, clearer comparative view.
- Advanced Visual Metrics: Expect to encounter a broader range of visual metrics, each designed to provide deeper insights into the model's training and validation performance.

Numerical Output:



```
[*******************100%%**********************]  1 of 1 completed
1/1 [==============================] - 0s 139ms/step
Date: 2023-08-17, Predicted Price: 230.64585876464844
1/1 [==============================] - 0s 65ms/step
Date: 2023-08-18, Predicted Price: 229.2869415283203
1/1 [==============================] - 0s 122ms/step
Date: 2023-08-19, Predicted Price: 229.15982055664062
1/1 [==============================] - 0s 83ms/step
Date: 2023-08-20, Predicted Price: 229.14120483398438
1/1 [==============================] - 0s 182ms/step
Date: 2023-08-21, Predicted Price: 229.05735778808594
1/1 [==============================] - 0s 184ms/step
Date: 2023-08-22, Predicted Price: 228.95957946777344
1/1 [==============================] - 0s 151ms/step
Date: 2023-08-23, Predicted Price: 228.90367126464844
1/1 [==============================] - 0s 40ms/step
Date: 2023-08-24, Predicted Price: 228.91392517089844
1/1 [==============================] - 0s 38ms/step
Date: 2023-08-25, Predicted Price: 228.99127197265625
1/1 [==============================] - 0s 40ms/step
Date: 2023-08-26, Predicted Price: 229.12530517578125
```

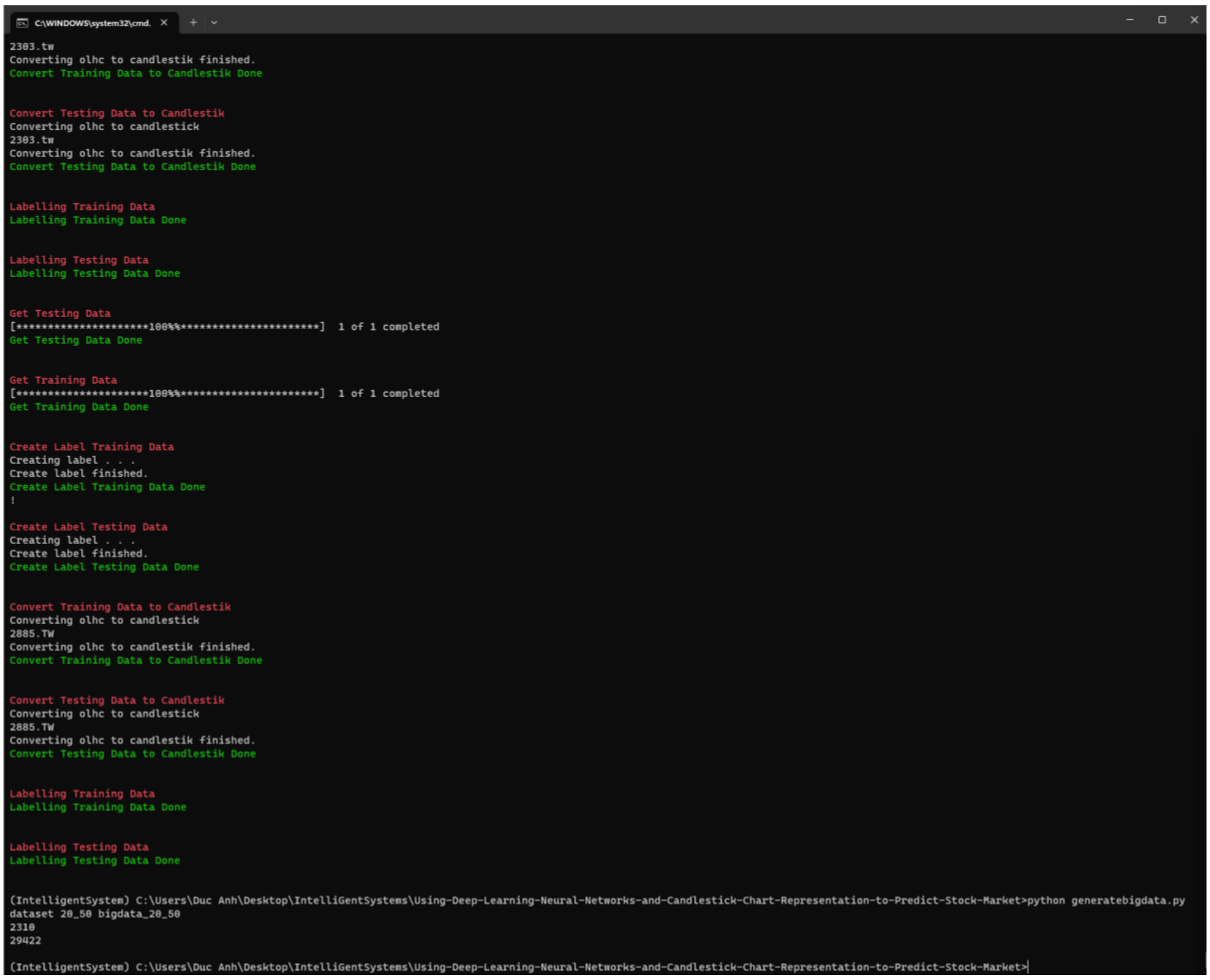
Figure P1.7.1 - Numerical Output for 10 upcoming days

- Advanced Model Evaluation Metrics: Dive into a richer set of evaluation metrics, offering a multi-dimensional view of the model's prediction prowess.
- Future Stock Price Predictions: The predictions for future stock prices are more detailed, potentially offering a breakdown by company or other criteria.

Extension Task- P2

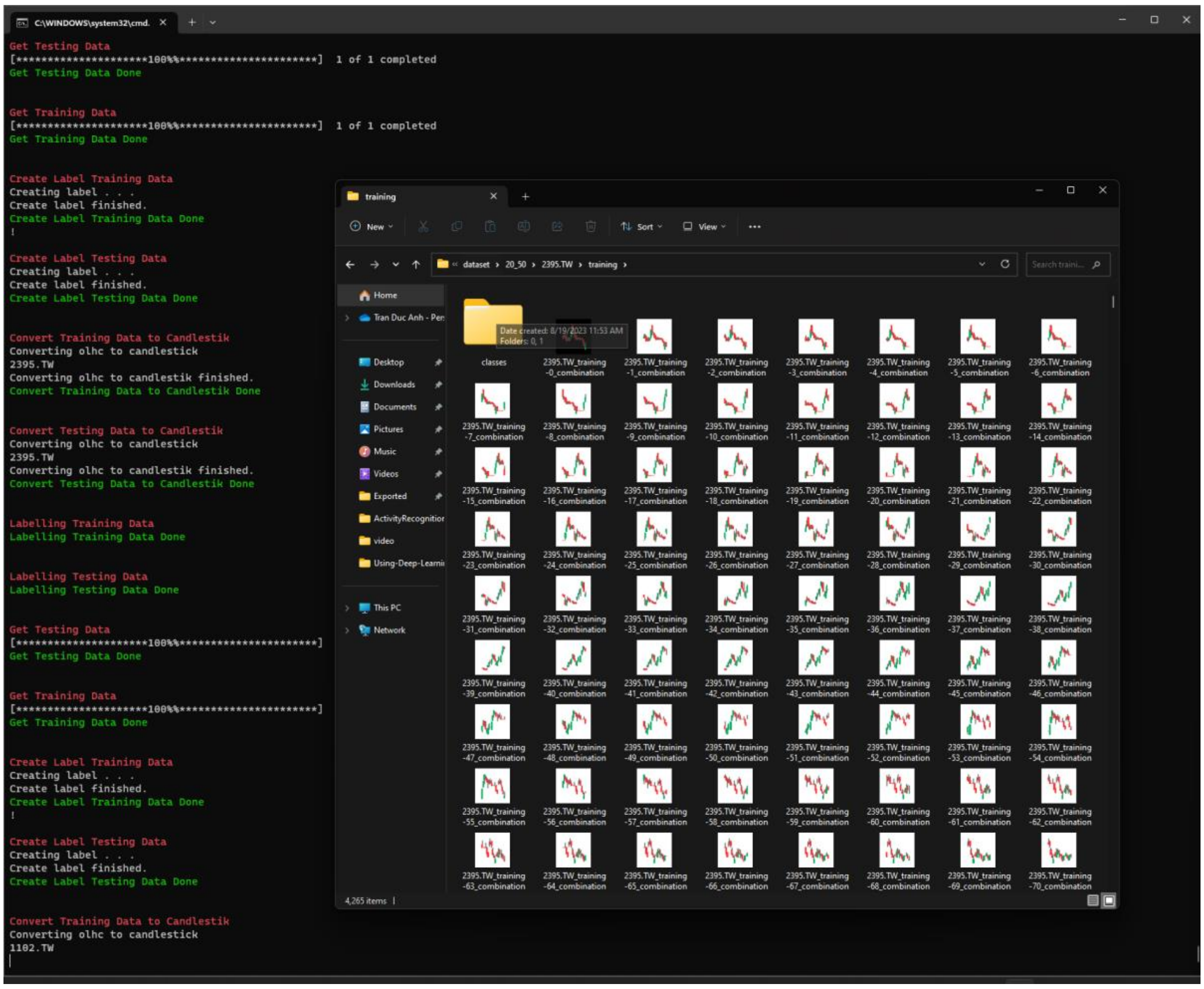
In this task, as the provided code is way too outdated and there are many data file is not yet been provided or it is missing, it is extremely hard to have this fixed. I have gone through the code, and I have founded a list of decencies that required to run this project, which I have listed above in the **Requirements** section.

I've been able to do the first step in Prepare dataset which is required to run this command:
python runallfromlist.py tw50.csv 20 50



```
2303.tw
Converting olhc to candlestik finished.
Convert Training Data to Candlestik Done


Convert Testing Data to Candlestik
Converting olhc to candlestick
2303.tw
Converting olhc to candlestik finished.
Convert Testing Data to Candlestik Done


Labelling Training Data
Labelling Training Data Done


Labelling Testing Data
Labelling Testing Data Done


Get Testing Data
[*********************100%%***********************]  1 of 1 completed
Get Testing Data Done


Get Training Data
[*********************100%%***********************]  1 of 1 completed
Get Training Data Done


Create Label Training Data
Creating label . . .
Create label finished.
Create Label Training Data Done
!

Create Label Testing Data
Creating label . . .
Create label finished.
Create Label Testing Data Done


Convert Training Data to Candlestik
Converting olhc to candlestick
2885.TW
Converting olhc to candlestik finished.
Convert Training Data to Candlestik Done


Convert Testing Data to Candlestik
Converting olhc to candlestick
2885.TW
Converting olhc to candlestik finished.
Convert Testing Data to Candlestik Done


Labelling Training Data
Labelling Training Data Done


Labelling Testing Data
Labelling Testing Data Done


(IntelligentSystem) C:\Users\Duc Anh\Desktop\IntelliGentSystems\Using-Deep-Learning-Neural-Networks-and-Candlestick-Chart-Representation-to-Predict-Stock-Market>python generatebigdata.py
dataset 20_50 bigdata_20_50
2310
29422

(IntelligentSystem) C:\Users\Duc Anh\Desktop\IntelliGentSystems\Using-Deep-Learning-Neural-Networks-and-Candlestick-Chart-Representation-to-Predict-Stock-Market>
```

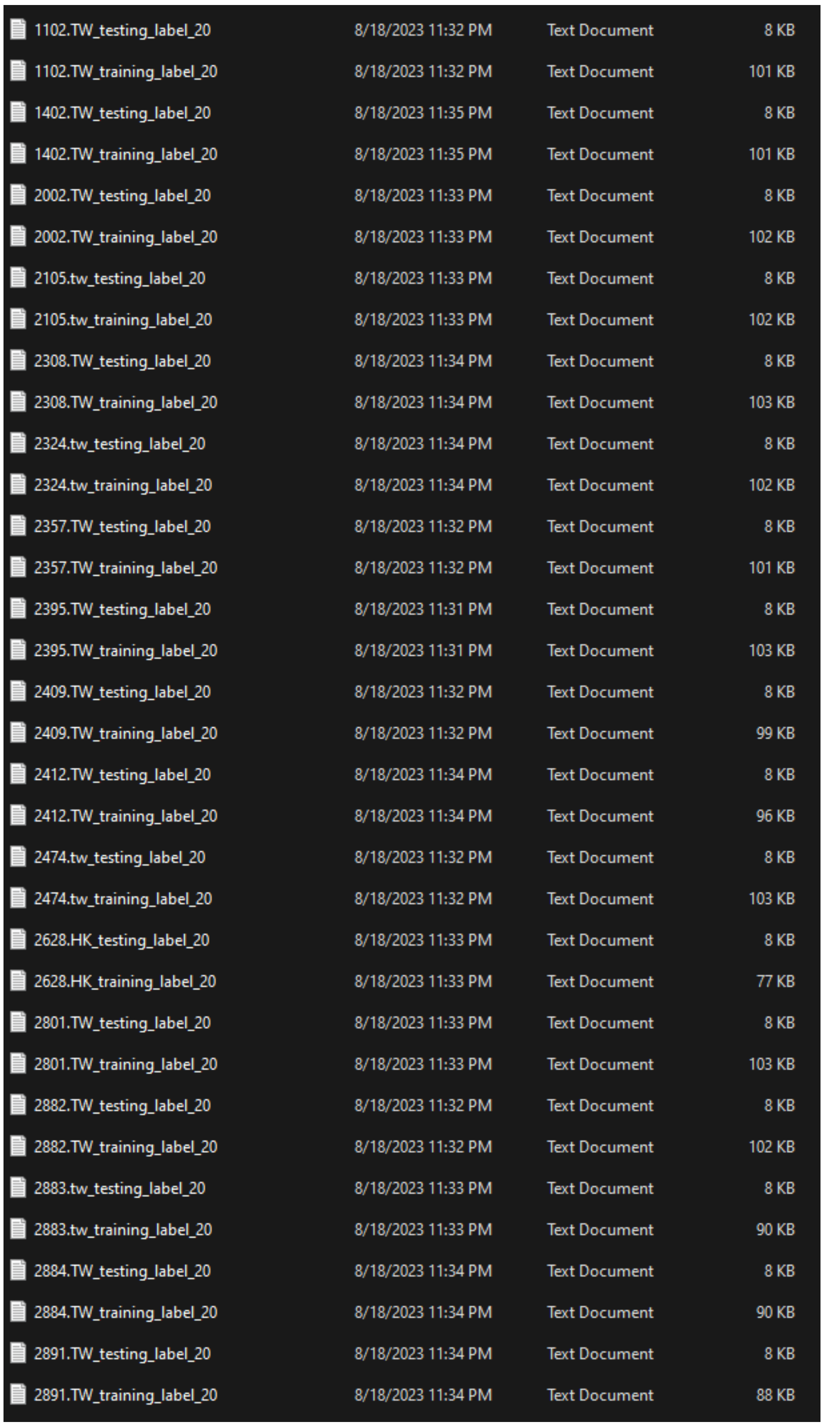
Figure P1.8.1 - Output python runallfromlist.py tw50.csv 20 50



```
Get Testing Data
[********************100%%**********************]  1 of 1 completed
Get Testing Data Done


Get Training Data
[********************100%%**********************]  1 of 1 completed
Get Training Data Done


Create Label Training Data
Creating label . . .
Create label finished.
Create Label Training Data Done
!

Create Label Testing Data
Creating label . . .
Create label finished.
Create Label Testing Data Done


Convert Training Data to Candlestik
Converting olhc to candlestick
2395.TW
Converting olhc to candlestik finished.
Convert Training Data to Candlestik Done


Convert Testing Data to Candlestik
Converting olhc to candlestick
2395.TW
Converting olhc to candlestik finished.
Convert Testing Data to Candlestik Done


Labelling Training Data
Labelling Training Data Done


Labelling Testing Data
Labelling Testing Data Done


Get Testing Data
[********************100%%**********************]
Get Testing Data Done


Get Training Data
[********************100%%**********************]
Get Training Data Done


Create Label Training Data
Creating label . . .
Create label finished.
Create Label Training Data Done
!

Create Label Testing Data
Creating label . . .
Create label finished.
Create Label Testing Data Done


Convert Training Data to Candlestik
Converting olhc to candlestick
1102.TW
|
```

Figure P1.8.2 - Output python runallfromlist.py tw50.csv 20 50 files

In file **runallfromlist.py,** which will link to **prepprocess_binclass.py** and **getdata.py.** There are a few things that need to be modified such as updating the libraries to the newest version as the older one is no longer supported. Once it run, there are a few file that will be generated in the screenshot down below.



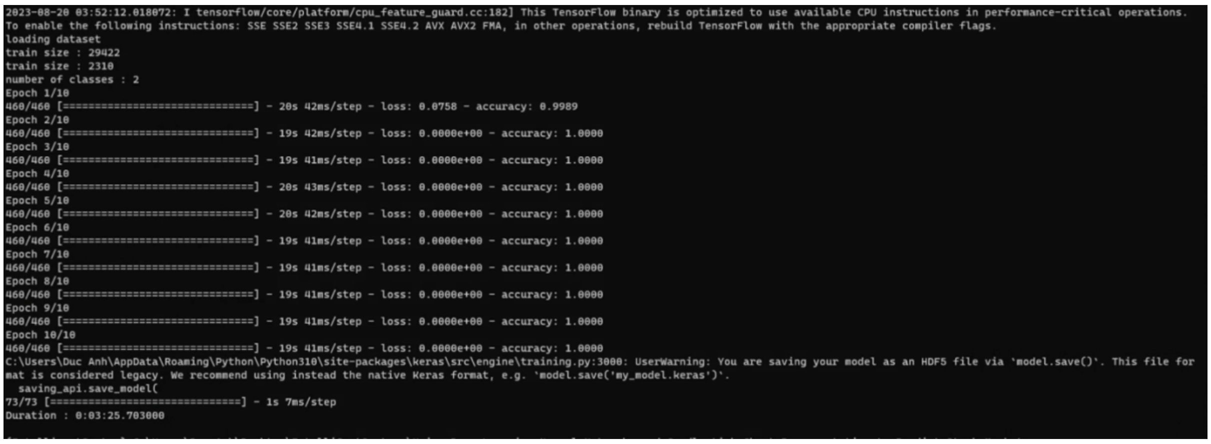
```
1102.TW_testing_label_20      8/18/2023 11:32 PM   Text Document     8 KB
1102.TW_training_label_20     8/18/2023 11:32 PM   Text Document   101 KB
1402.TW_testing_label_20      8/18/2023 11:35 PM   Text Document     8 KB
1402.TW_training_label_20     8/18/2023 11:35 PM   Text Document   101 KB
2002.TW_testing_label_20      8/18/2023 11:33 PM   Text Document     8 KB
2002.TW_training_label_20     8/18/2023 11:33 PM   Text Document   102 KB
2105.tw_testing_label_20      8/18/2023 11:33 PM   Text Document     8 KB
2105.tw_training_label_20     8/18/2023 11:33 PM   Text Document   102 KB
2308.TW_testing_label_20      8/18/2023 11:34 PM   Text Document     8 KB
2308.TW_training_label_20     8/18/2023 11:34 PM   Text Document   103 KB
2324.tw_testing_label_20      8/18/2023 11:34 PM   Text Document     8 KB
2324.tw_training_label_20     8/18/2023 11:34 PM   Text Document   102 KB
2357.TW_testing_label_20      8/18/2023 11:32 PM   Text Document     8 KB
2357.TW_training_label_20     8/18/2023 11:32 PM   Text Document   101 KB
2395.TW_testing_label_20      8/18/2023 11:31 PM   Text Document     8 KB
2395.TW_training_label_20     8/18/2023 11:31 PM   Text Document   103 KB
2409.TW_testing_label_20      8/18/2023 11:32 PM   Text Document     8 KB
2409.TW_training_label_20     8/18/2023 11:32 PM   Text Document    99 KB
2412.TW_testing_label_20      8/18/2023 11:34 PM   Text Document     8 KB
2412.TW_training_label_20     8/18/2023 11:34 PM   Text Document    96 KB
2474.tw_testing_label_20      8/18/2023 11:32 PM   Text Document     8 KB
2474.tw_training_label_20     8/18/2023 11:32 PM   Text Document   103 KB
2628.HK_testing_label_20      8/18/2023 11:33 PM   Text Document     8 KB
2628.HK_training_label_20     8/18/2023 11:33 PM   Text Document    77 KB
2801.TW_testing_label_20      8/18/2023 11:33 PM   Text Document     8 KB
2801.TW_training_label_20     8/18/2023 11:33 PM   Text Document   103 KB
2882.TW_testing_label_20      8/18/2023 11:32 PM   Text Document     8 KB
2882.TW_training_label_20     8/18/2023 11:32 PM   Text Document   102 KB
2883.tw_testing_label_20      8/18/2023 11:33 PM   Text Document     8 KB
2883.tw_training_label_20     8/18/2023 11:33 PM   Text Document    90 KB
2884.TW_testing_label_20      8/18/2023 11:34 PM   Text Document     8 KB
2884.TW_training_label_20     8/18/2023 11:34 PM   Text Document    90 KB
2891.TW_testing_label_20      8/18/2023 11:34 PM   Text Document     8 KB
2891.TW_training_label_20     8/18/2023 11:34 PM   Text Document    88 KB
```

Figure P1.8.3 - Output python runallfromlist.py tw50.csv 20 50

When running this command, after I modified a few codes. There are some errors showing up especially in **Convert Testing Data to Candlestick,** I believe that this problem arises from using the older **mplfinance** plotting functions with the newer version of **matploitlib**.

In myDeepCNN.py, looks like it used the **Tensorflow 1.x.x** which has been depreciated therefore **tf.ConfigProto()** is no longer available in **Tensorflow 2.x.x.** Some of the **Keras** also outdated and needs to be update. This is just a small part, as there are so many more that needs to be updated in this file.

After I re-coded so it suits the newer version, here is the final output of the program.



```
2023-08-20 03:52:12.018072: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
loading dataset
train size : 29422
train size : 2310
number of classes : 2
Epoch 1/10
460/460 [==============================] - 20s 42ms/step - loss: 0.0758 - accuracy: 0.9989
Epoch 2/10
460/460 [==============================] - 19s 42ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 3/10
460/460 [==============================] - 19s 41ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 4/10
460/460 [==============================] - 20s 43ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 5/10
460/460 [==============================] - 20s 42ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 6/10
460/460 [==============================] - 19s 41ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 7/10
460/460 [==============================] - 19s 41ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 8/10
460/460 [==============================] - 19s 41ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 9/10
460/460 [==============================] - 19s 41ms/step - loss: 0.0000e+00 - accuracy: 1.0000
Epoch 10/10
460/460 [==============================] - 19s 41ms/step - loss: 0.0000e+00 - accuracy: 1.0000
C:\Users\Duc Anh\AppData\Roaming\Python\Python310\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file for
mat is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
73/73 [==============================] - 1s 7ms/step
Duration : 0:03:25.703000
```

Figure P1.8.4 - Train Model