COS30018 – Intelligent System

Option B: Stock Price Prediction

Report P1

Name Phong Tran
Student Id:104334842

Class: 1-5
Tutor: Tuan Dung Lai

Table of Contents

Introduction

This report is based on the testing the code base of two folder v.01 and P1 all from GITHUB, with the guide of how to understand the code of folder v0.1 from YouTube, then running the code with the support of virtual environments. Finally, displaying the plots, which show the graph of two companies: Meta and CBA.AX.

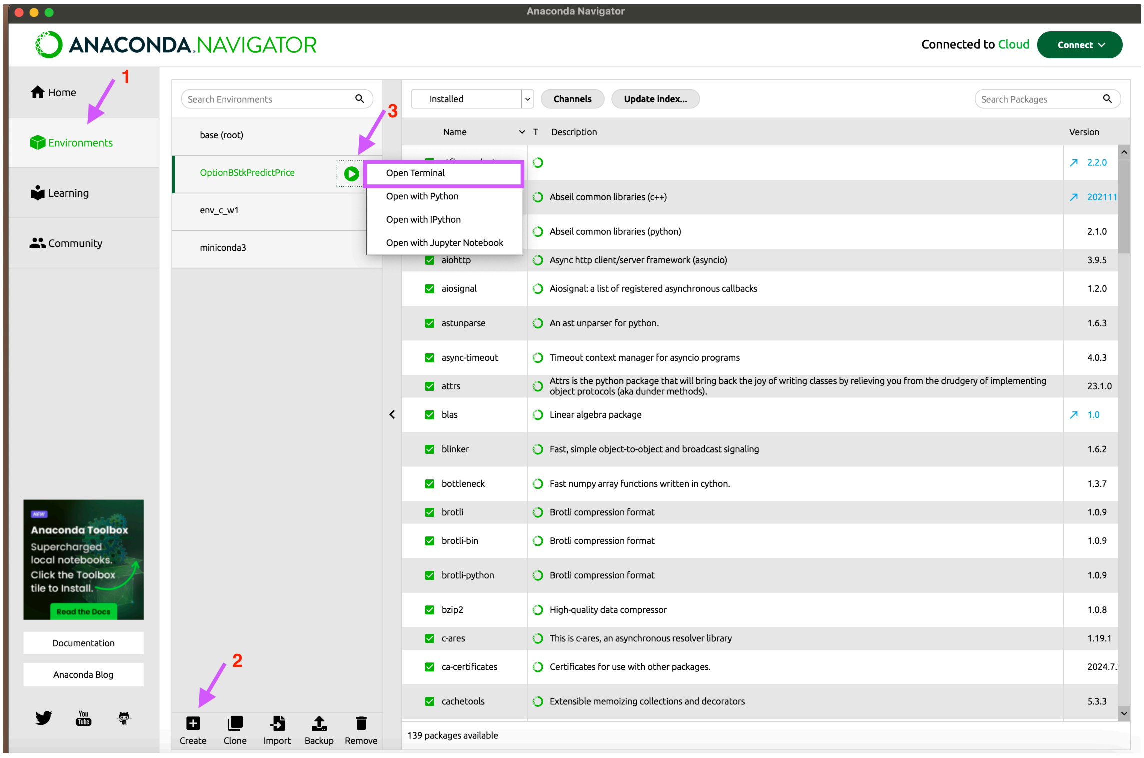
Setup the environment

There are two step that we need to do, first setup the virtual environment at Anaconda-navigator. Secondly, setup project at Pycharm and integrate with our own virtual environment that we have created before.

Setup Virtual Environment

Using Anaconda to setup virtual environment.

1. Download the Anaconda navigator
2. Setup the environment with the instructions

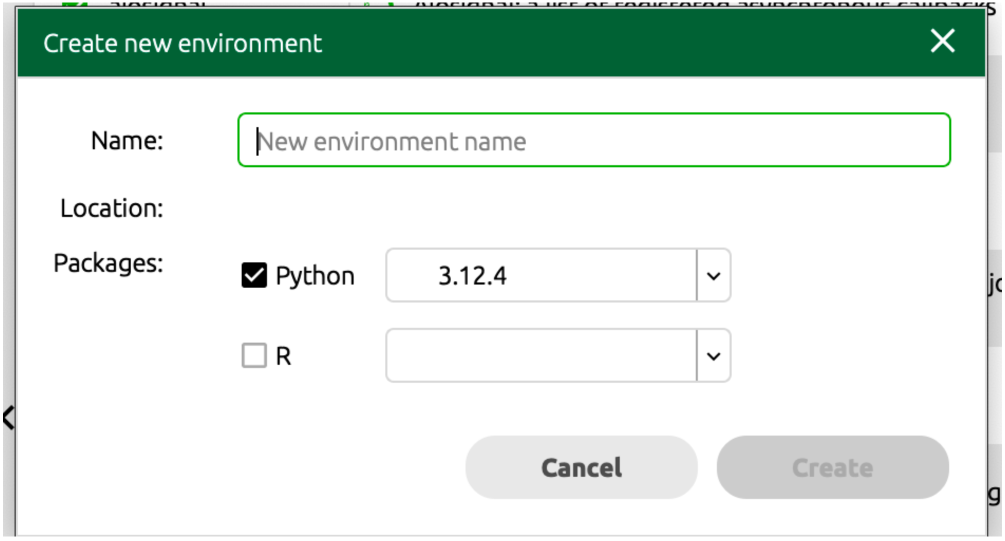


Name	T	Description	Version
	○		2.2.0
	○	Abseil common libraries (c++)	202111
	○	Abseil common libraries (python)	2.1.0
☑ aiohttp	○	Async http client/server framework (asyncio)	3.9.5
☑ aiosignal	○	Aiosignal: a list of registered asynchronous callbacks	1.2.0
☑ astunparse	○	An ast unparser for python.	1.6.3
☑ async-timeout	○	Timeout context manager for asyncio programs	4.0.3
☑ attrs	○	Attrs is the python package that will bring back the joy of writing classes by relieving you from the drudgery of implementing object protocols (aka dunder methods).	23.1.0
☑ blas	○	Linear algebra package	1.0
☑ blinker	○	Fast, simple object-to-object and broadcast signaling	1.6.2
☑ bottleneck	○	Fast numpy array functions written in cython.	1.3.7
☑ brotli	○	Brotli compression format	1.0.9
☑ brotli-bin	○	Brotli compression format	1.0.9
☑ brotli-python	○	Brotli compression format	1.0.9
☑ bzip2	○	High-quality data compressor	1.0.8
☑ c-ares	○	This is c-ares, an asynchronous resolver library	1.19.1
☑ ca-certificates	○	Certificates for use with other packages.	2024.7.
☑ cachetools	○	Extensible memoizing collections and decorators	5.3.3

139 packages available

The image shows instructions on how to install the libraries

At 2, when create a new environment, we should choose to create with latest python version before clicking the Create button



Name: New environment name

Location:

Packages: ☑ Python 3.12.4

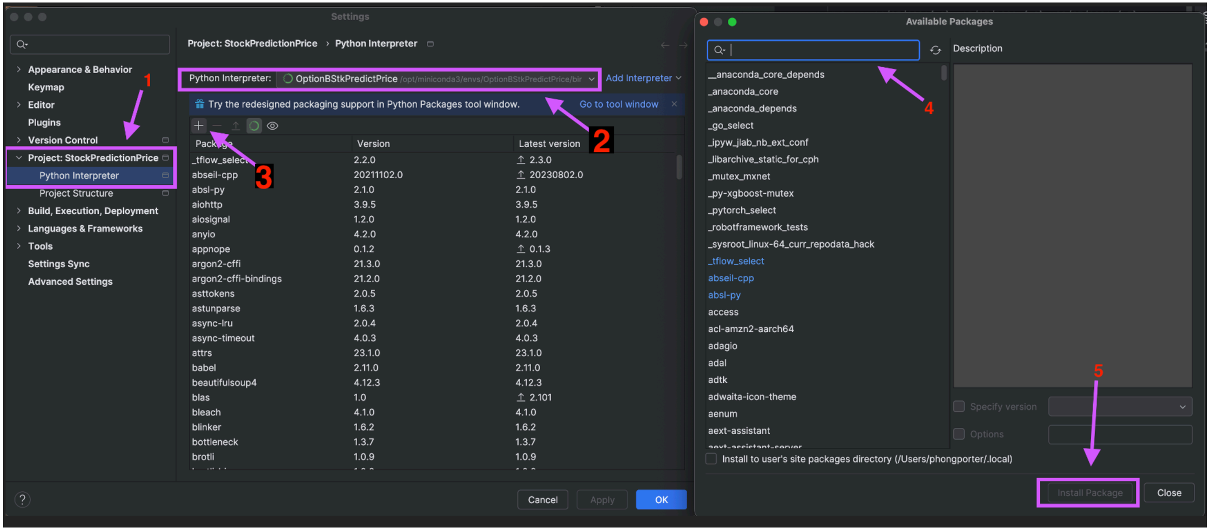
□ R

3. Install the required library

First of all, we need to install pip with command below: `conda install pip`. After that we are able to download any libraries required below, which from the code at v0.1

```
# Need to install the following (best in a virtual env)
# pip install numpy
# pip install matplotlib
# pip install pandas
# pip install tensorflow
# pip install scikit-learn
# pip install pandas-datareader
# pip install yfinance
```

Another choice to install the libraries is to go to Python interpreter, then choose the virtual environment that we have been created before. After that, click the + icon to open Available Packages window, then search the available libraries (some libraries might not be available due to limitation at Pycharm). Next, click Install Package button to install the package that you find. Finally at setting, click OK button to finish.



Python Interpreter: OptionBStkPredictPrice /opt/miniconda3/envs/OptionBStkPredictPrice/bir

Try the redesigned packaging support in Python Packages tool window. Go to tool window

Package	Version	Latest version
_tflow_select	2.2.0	↑ 2.3.0
abseil-cpp	20211102.0	↑ 20230802.0
absl-py	2.1.0	2.1.0
aiohttp	3.9.5	3.9.5
aiosignal	1.2.0	1.2.0
anyio	4.2.0	4.2.0
appnope	0.1.2	↑ 0.1.3
argon2-cffi	21.3.0	21.3.0
argon2-cffi-bindings	21.2.0	21.2.0
asttokens	2.0.5	2.0.5
astunparse	1.6.3	1.6.3
async-lru	2.0.4	2.0.4
async-timeout	4.0.3	4.0.3
attrs	23.1.0	23.1.0
babel	2.11.0	2.11.0
beautifulsoup4	4.12.3	4.12.3
blas	1.0	↑ 2.101
bleach	4.1.0	4.1.0
blinker	1.6.2	1.6.2
bottleneck	1.3.7	1.3.7
brotli	1.0.9	1.0.9

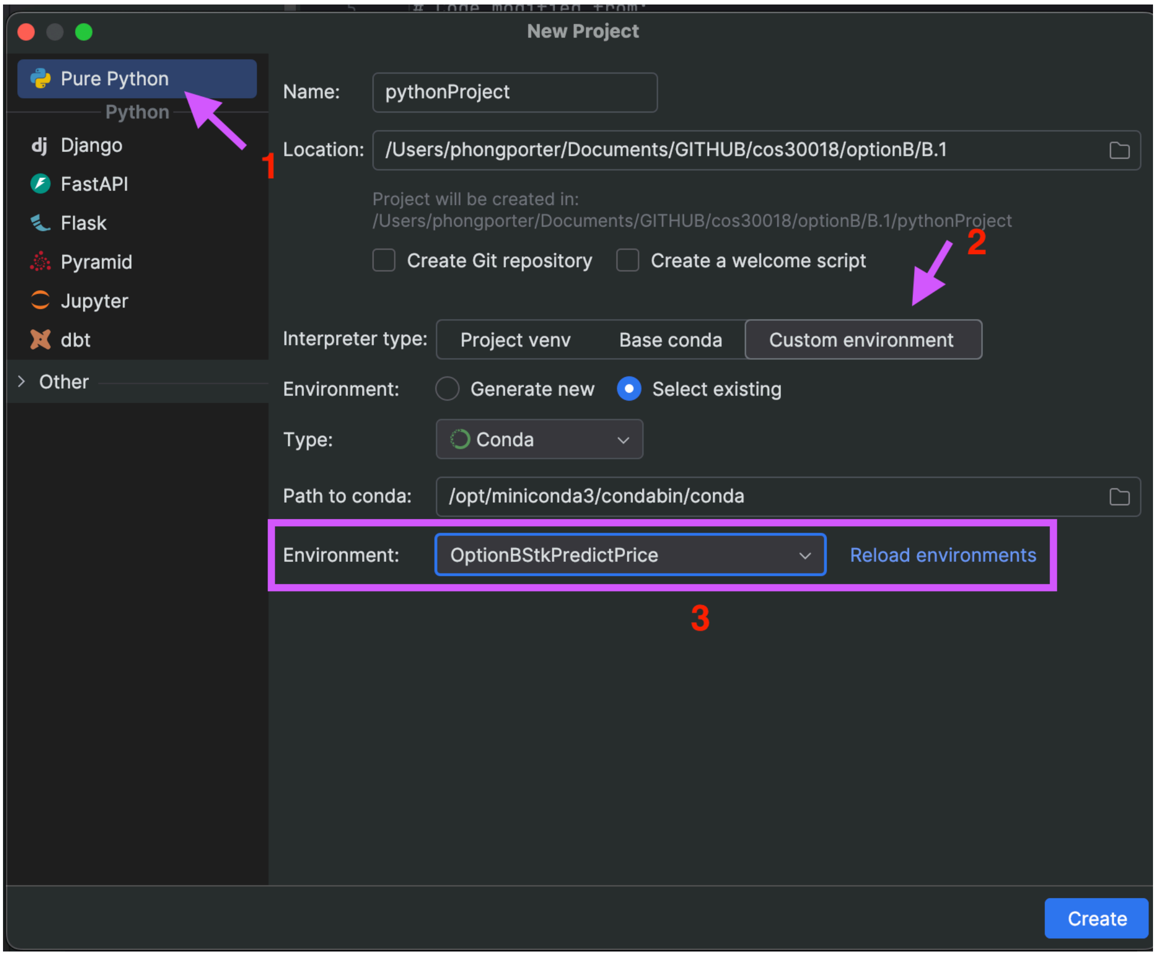
__anaconda_core_depends
_anaconda_core
_anaconda_depends
_go_select
_ipyw_jlab_nb_ext_conf
_libarchive_static_for_cph
_mutex_mxnet
_py-xgboost-mutex
_pytorch_select
_robotframework_tests
_sysroot_linux-64_curr_repodata_hack
_tflow_select
abseil-cpp
absl-py
access
acl-amzn2-aarch64
adagio
adal
adtk
adwaita-icon-theme
aenum
aext-assistant
aext-assistant-server

☐ Install to user's site packages directory (/Users/phongporter/.local)

Setup Project

I use Pycharm to create a new project because it an IDE tool for Python, specifically for data science and machine learning.

Click Pure Python to create project using Python. Then choose custom environment with Select existing at Environment after we create successfully our own environment at Anaconda-navigator (we can reload environment if it does not display). Finally, click Create button to create the project. (The image below)



New Project

Name: pythonProject

Location: /Users/phongporter/Documents/GITHUB/cos30018/optionB/B.1

Project will be created in: /Users/phongporter/Documents/GITHUB/cos30018/optionB/B.1/pythonProject

□ Create Git repository □ Create a welcome script

Interpreter type: Project venv | Base conda | Custom environment

Environment: ○ Generate new ● Select existing

Type: Conda

Path to conda: /opt/miniconda3/condabin/conda

Environment: OptionBStkPredictPrice

Instruction on how to install the project with instruction on the image (sketched)

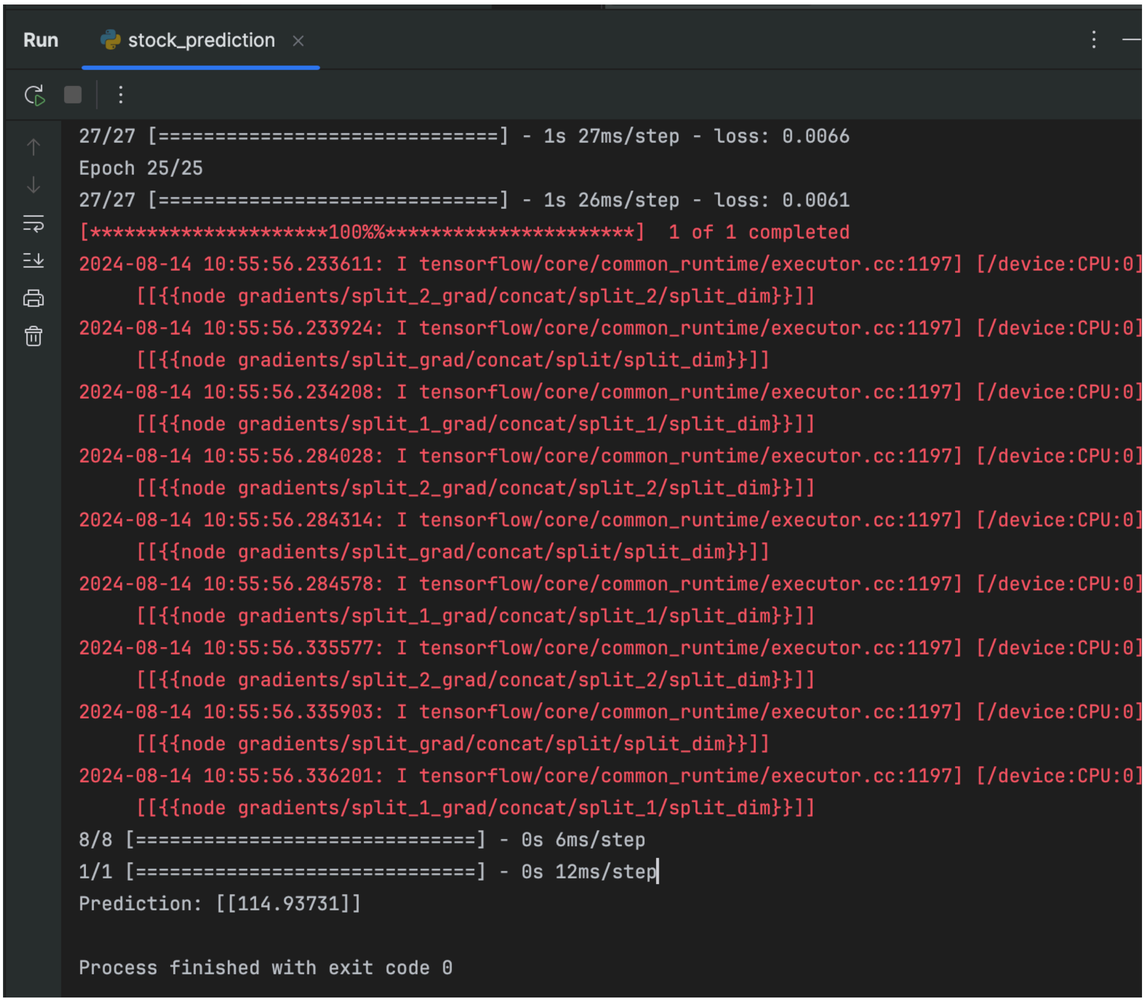
Code base testing

In this work, I will test two code bases, which are v0.1 and P1, in order to find out whether run successfully or not.

v0.1

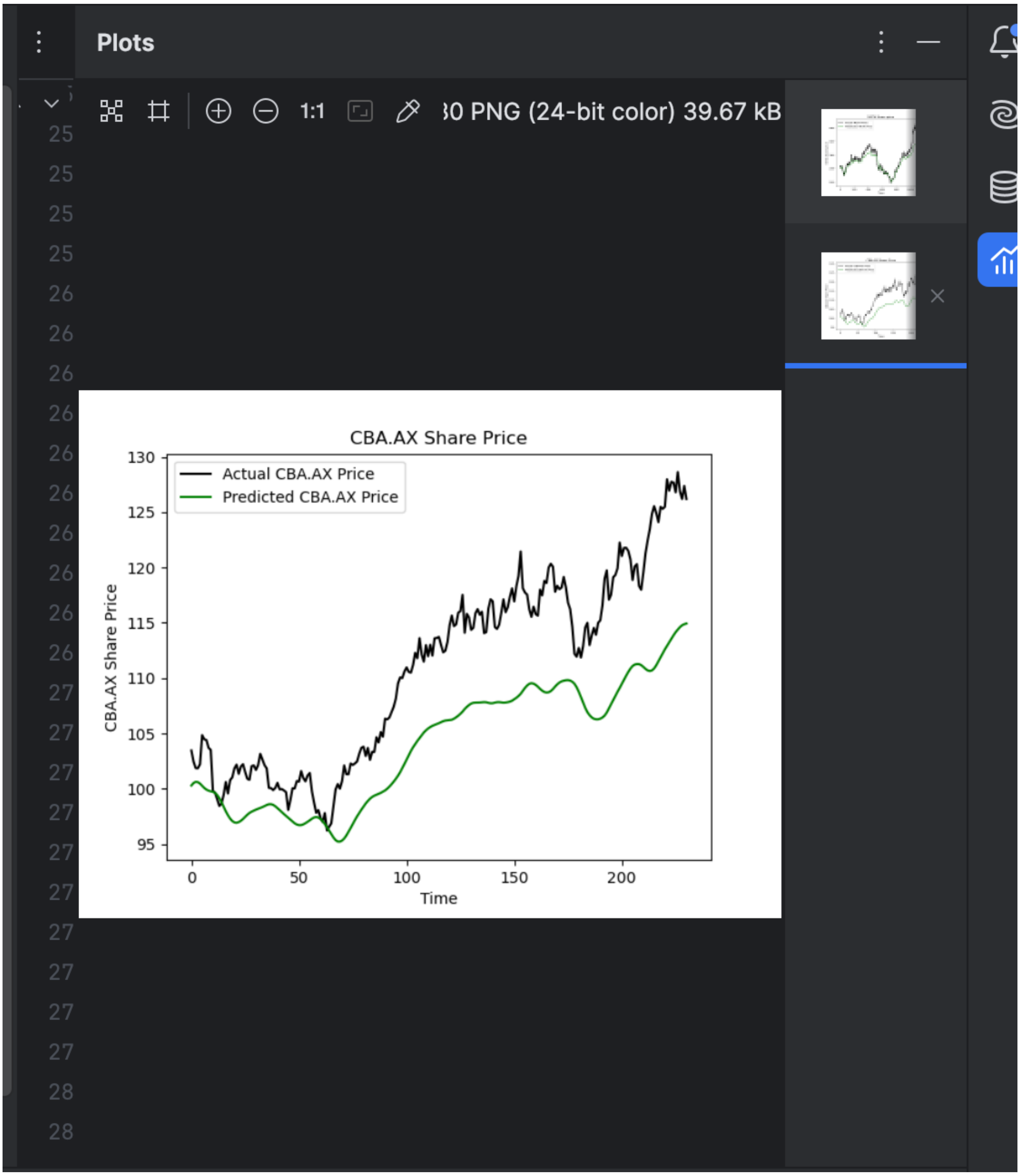
When I download the stock_prediction.py, I need to create new directory at project directory to make sure the folder directory could split among two code bases: v0.1 and P1.

After putting the code into the directory v0.1, I click run icon at built in Pycharm to run code. Luckily, it work without any issues. The running console shows the model training in the dataset from the external source(epoch)



```
27/27 [==============================] - 1s 27ms/step - loss: 0.0066
Epoch 25/25
27/27 [==============================] - 1s 26ms/step - loss: 0.0061
[*********************100%***********************]  1 of 1 completed
2024-08-14 10:55:56.233611: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_2_grad/concat/split_2/split_dim}}]]
2024-08-14 10:55:56.233924: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_grad/concat/split/split_dim}}]]
2024-08-14 10:55:56.234208: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_1_grad/concat/split_1/split_dim}}]]
2024-08-14 10:55:56.284028: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_2_grad/concat/split_2/split_dim}}]]
2024-08-14 10:55:56.284314: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_grad/concat/split/split_dim}}]]
2024-08-14 10:55:56.284578: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_1_grad/concat/split_1/split_dim}}]]
2024-08-14 10:55:56.335577: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_2_grad/concat/split_2/split_dim}}]]
2024-08-14 10:55:56.335903: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_grad/concat/split/split_dim}}]]
2024-08-14 10:55:56.336201: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0]
     [[{{node gradients/split_1_grad/concat/split_1/split_dim}}]]
8/8 [==============================] - 0s 6ms/step
1/1 [==============================] - 0s 12ms/step
Prediction: [[114.93731]]

Process finished with exit code 0
```

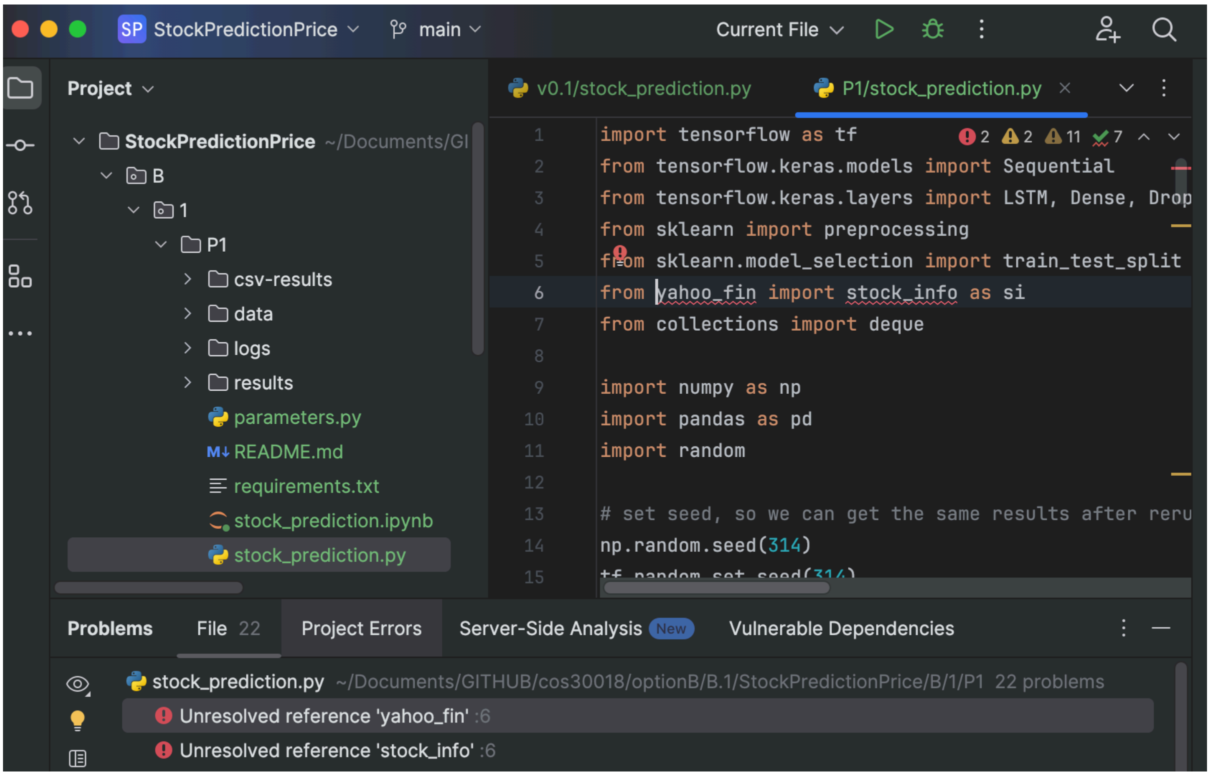
Training data successfully, with the Prediction display the following day



Click the Plots to see the image, which is the result of the code base.

P1

After setup the code base from GitHub, which I have to fork the repo from the GitHub before take only the stock-prediction folder.



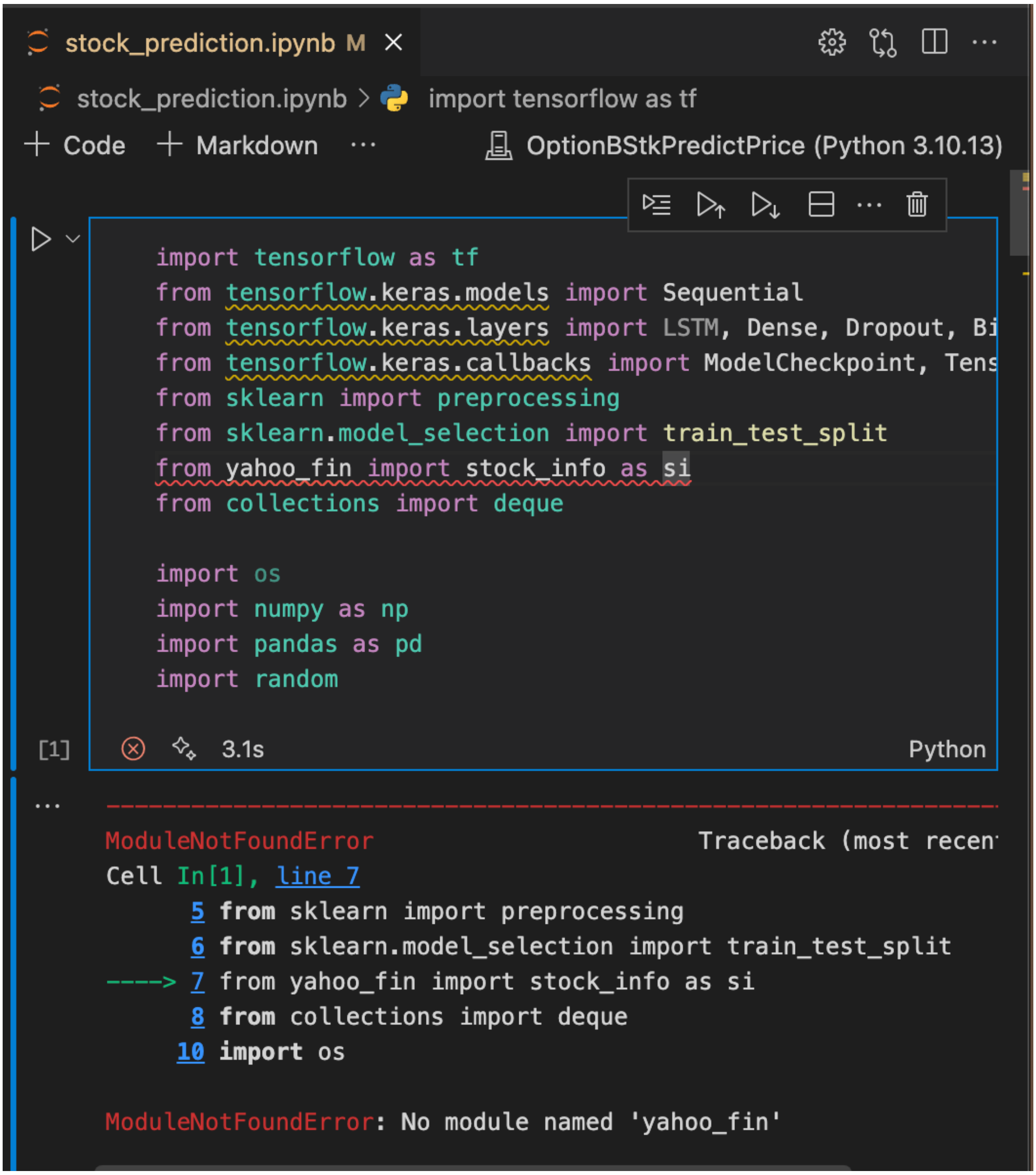
```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Drop
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from yahoo_fin import stock_info as si
from collections import deque

import numpy as np
import pandas as pd
import random

# set seed, so we can get the same results after reru
np.random.seed(314)
tf.random.set_seed(314)
```

stock_prediction.py ~/Documents/GITHUB/cos30018/optionB/B.1/StockPredictionPrice/B/1/P1 22 problems

Unresolved reference 'yahoo_fin' :6

Unresolved reference 'stock_info' :6

When I open the code, I saw there are two bugs, which may be the reason of not install the library


stock_prediction.ipynb M
stock_prediction.ipynb > import tensorflow as tf
+ Code
+ Markdown
OptionBStkPredictPrice (Python 3.10.13)
[1]
3.1s
Python

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Bi
from tensorflow.keras.callbacks import ModelCheckpoint, Tens
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from yahoo_fin import stock_info as si
from collections import deque

import os
import numpy as np
import pandas as pd
import random
```

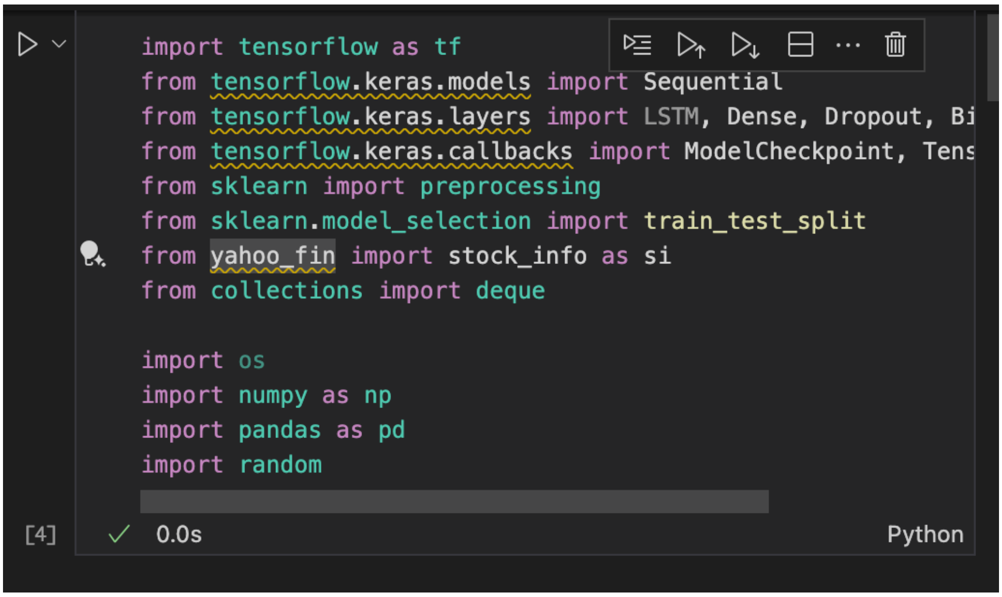
```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recen
Cell In[1], line 7
      5 from sklearn import preprocessing
      6 from sklearn.model_selection import train_test_split
----> 7 from yahoo_fin import stock_info as si
      8 from collections import deque
     10 import os

ModuleNotFoundError: No module named 'yahoo_fin'
```

Even when I run the file with ipynb extension, it still show the bugs which quite similar to the .py file

To resolve the issue, I have to install yahoo_fin library at terminal from the virtual environment that associated with the project codebases: ` pip install yahoo_fin`

After it install successfully, I have run the code at .ipynb and it said "`Certain functionality requires requests_html, which is not installed`". That means I have to install requests_html package.

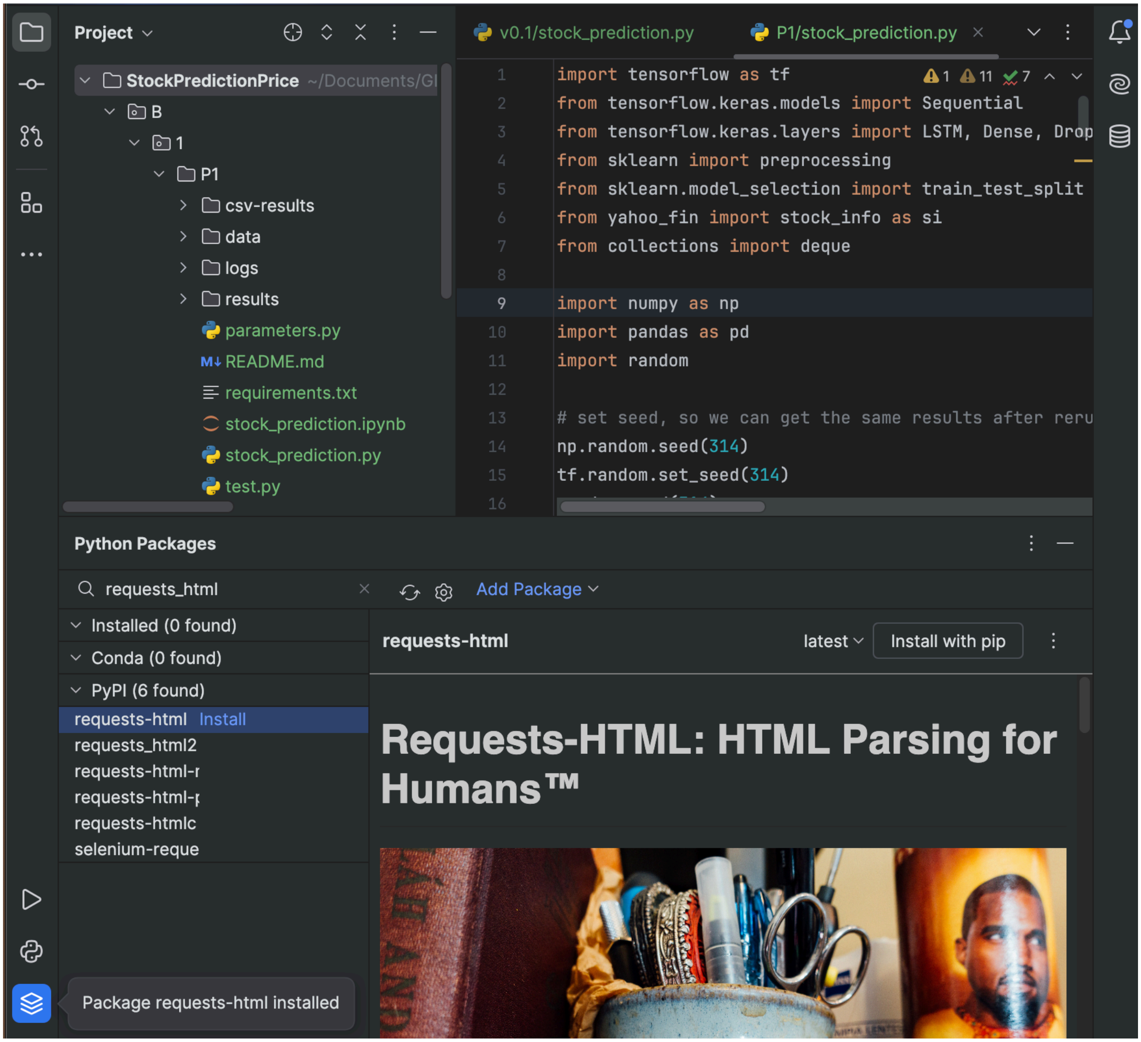
After install the requests_html package, I run the ipynb file and it shows successfully as it ticked green icon



```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Bi
from tensorflow.keras.callbacks import ModelCheckpoint, Tens
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from yahoo_fin import stock_info as si
from collections import deque

import os
import numpy as np
import pandas as pd
import random
```

```
[4]  ✓  0.0s                                                Python
```

1. Run the python file (.py) at Pycharm



```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Drop
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from yahoo_fin import stock_info as si
from collections import deque

import numpy as np
import pandas as pd
import random

# set seed, so we can get the same results after reru
np.random.seed(314)
tf.random.set_seed(314)
```

Python Packages

requests_html

Add Package

Installed (0 found)

Conda (0 found)

PyPI (6 found)

requests-html Install
requests_html2
requests-html-r
requests-html-p
requests-htmlc
selenium-reque

requests-html latest Install with pip

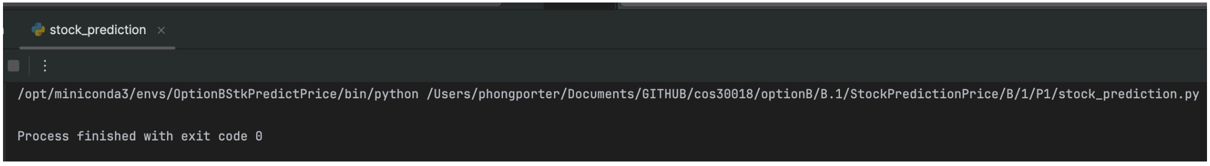
Requests-HTML: HTML Parsing for Humans™

Package requests-html installed

List of python packages as I click the button at the right bottom of the image

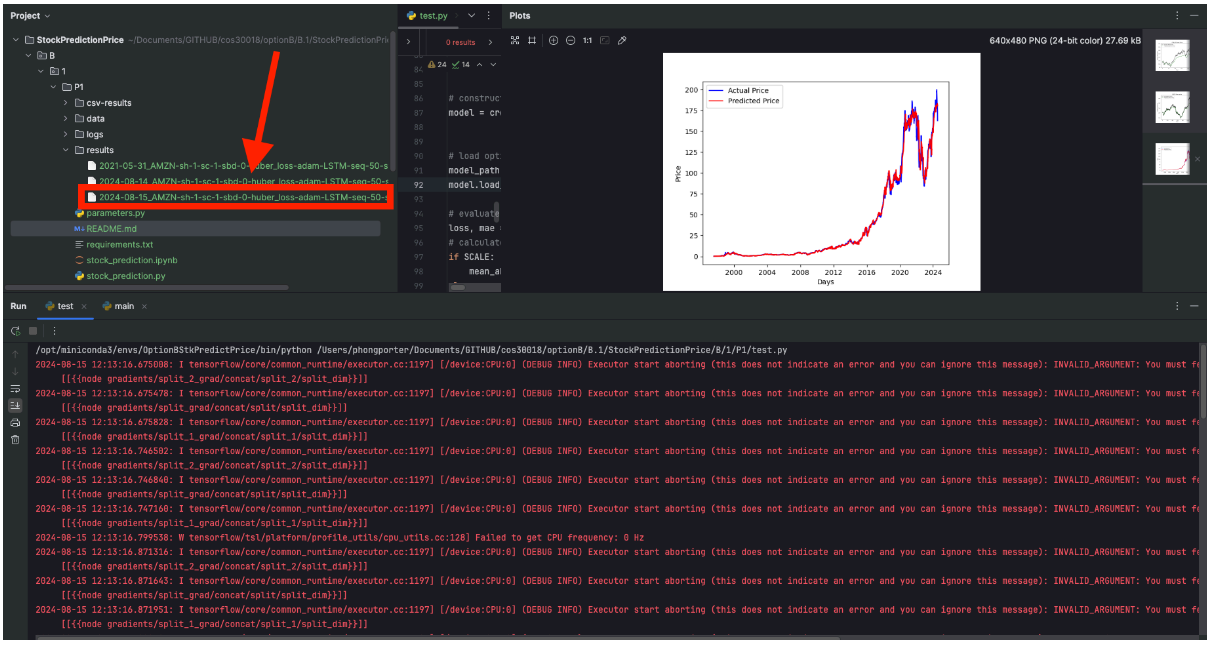
When I add another line of code: import requests_html, I run the code and it shows the ImportError code, so I have to install lxml_html_clean package

After installing, I run the stock_prediction.py and it show nothing without bugs or warning.



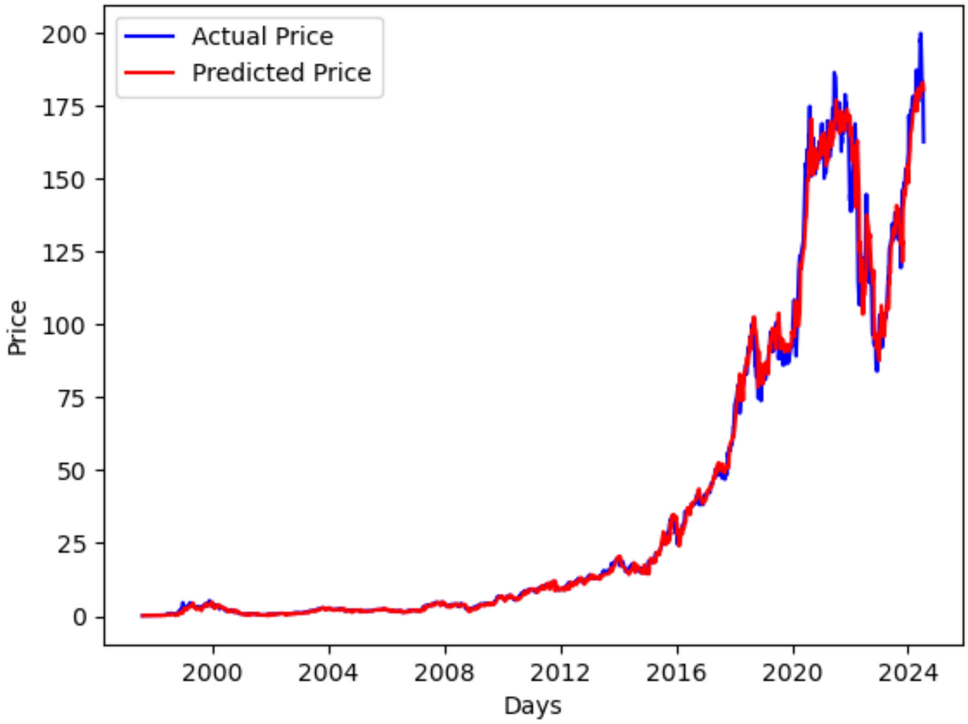
```
/opt/miniconda3/envs/OptionBStkPredictPrice/bin/python /Users/phongporter/Documents/GITHUB/cos30018/optionB/B.1/StockPredictionPrice/B/1/P1/stock_prediction.py

Process finished with exit code 0
```

Run the train.py to start training the parameters that specified. Then after successfully training (with generated latest file at results directory) at the image below

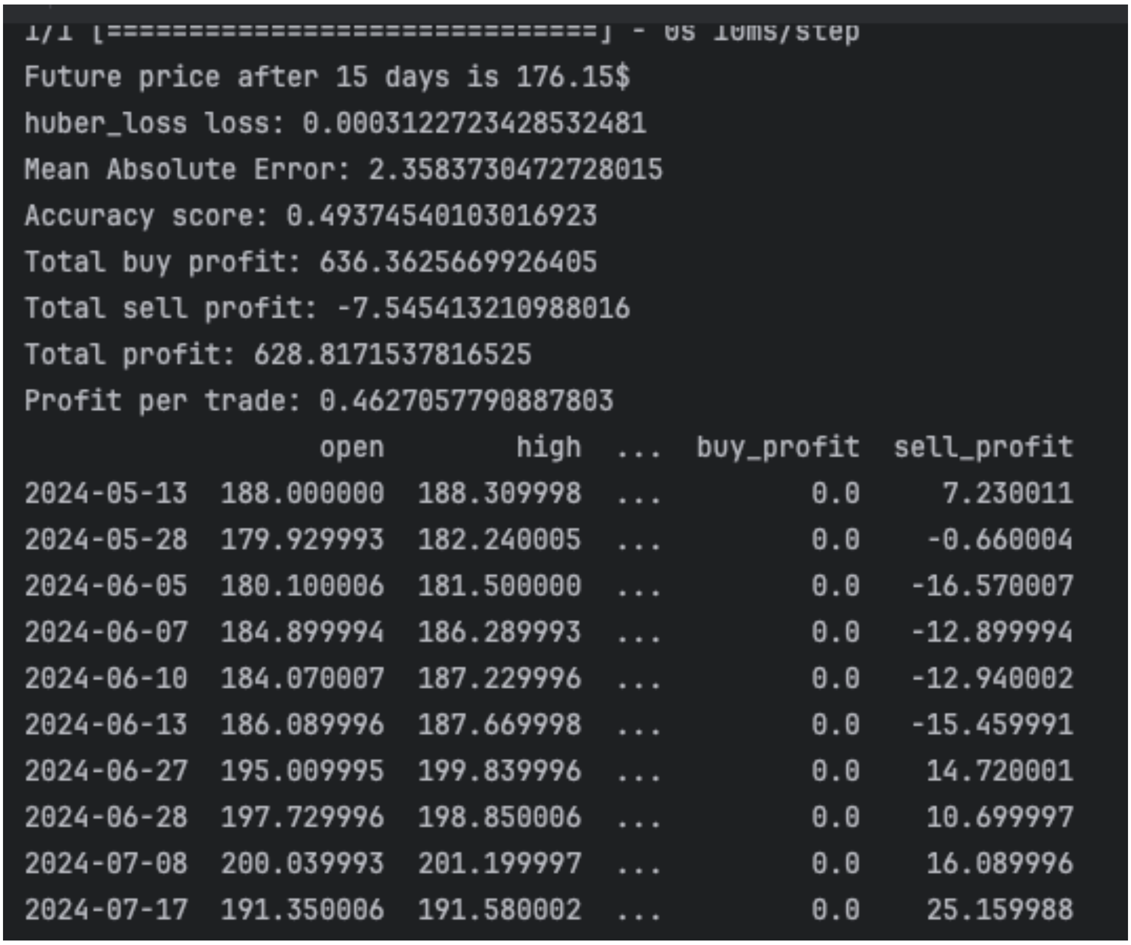


```
/opt/miniconda3/envs/OptionBStkPredictPrice/bin/python /Users/phongporter/Documents/GITHUB/cos30018/optionB/B.1/StockPredictionPrice/B/1/P1/test.py
2024-08-15 12:13:16.675008: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_2_grad/concat/split_2/split_dim}}]]
2024-08-15 12:13:16.675478: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_grad/concat/split/split_dim}}]]
2024-08-15 12:13:16.675828: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_1_grad/concat/split_1/split_dim}}]]
2024-08-15 12:13:16.746502: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_2_grad/concat/split_2/split_dim}}]]
2024-08-15 12:13:16.746840: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_grad/concat/split/split_dim}}]]
2024-08-15 12:13:16.747160: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_1_grad/concat/split_1/split_dim}}]]
2024-08-15 12:13:16.799538: W tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU frequency: 0 Hz
2024-08-15 12:13:16.871316: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_2_grad/concat/split_2/split_dim}}]]
2024-08-15 12:13:16.871643: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_grad/concat/split/split_dim}}]]
2024-08-15 12:13:16.871951: I tensorflow/core/common_runtime/executor.cc:1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not indicate an error and you can ignore this message): INVALID_ARGUMENT: You must fe
     [[{{node gradients/split_1_grad/concat/split_1/split_dim}}]]
```

Run test.py and it shows the plot, however, in order to make its python file run successfully, we must run the train.py to make it training and generate the new file at results folder if training successfully.



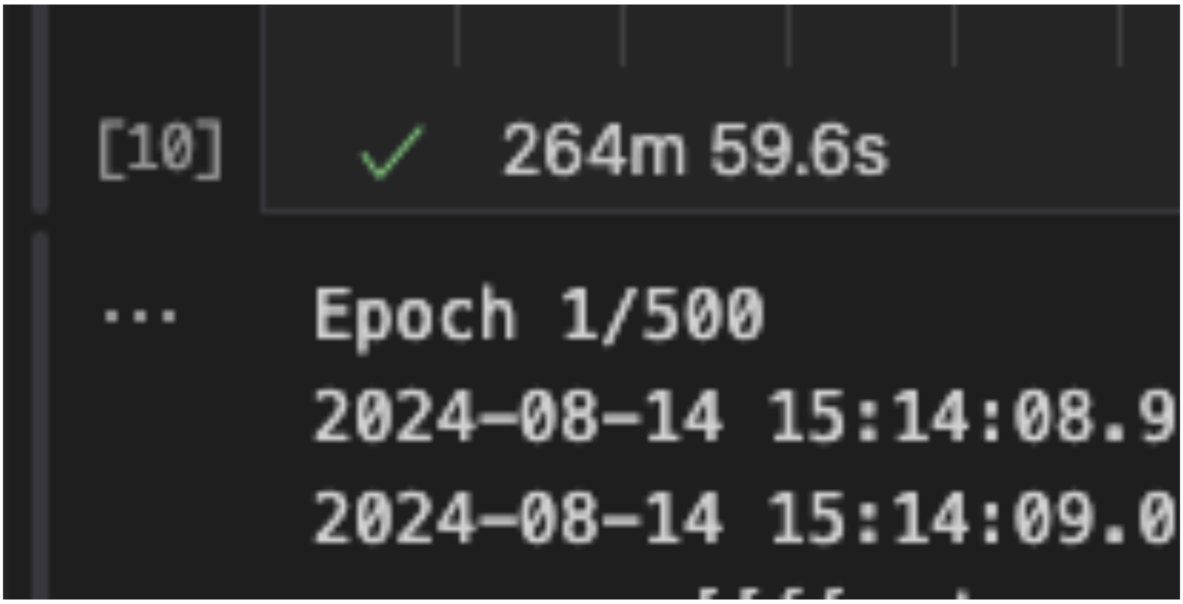
The Plot display successfully



```
1/1 [==============================] - 0s 10ms/step
Future price after 15 days is 176.15$
huber_loss loss: 0.0003122723428532481
Mean Absolute Error: 2.3583730472728015
Accuracy score: 0.49374540103016923
Total buy profit: 636.3625669926405
Total sell profit: -7.545413210988016
Total profit: 628.8171537816525
Profit per trade: 0.4627057790887803
                  open        high  ...  buy_profit  sell_profit
2024-05-13  188.000000  188.309998  ...         0.0     7.230011
2024-05-28  179.929993  182.240005  ...         0.0    -0.660004
2024-06-05  180.100006  181.500000  ...         0.0   -16.570007
2024-06-07  184.899994  186.289993  ...         0.0   -12.899994
2024-06-10  184.070007  187.229996  ...         0.0   -12.940002
2024-06-13  186.089996  187.669998  ...         0.0   -15.459991
2024-06-27  195.009995  199.839996  ...         0.0    14.720001
2024-06-28  197.729996  198.850006  ...         0.0    10.699997
2024-07-08  200.039993  201.199997  ...         0.0    16.089996
2024-07-17  191.350006  191.580002  ...         0.0    25.159988
```

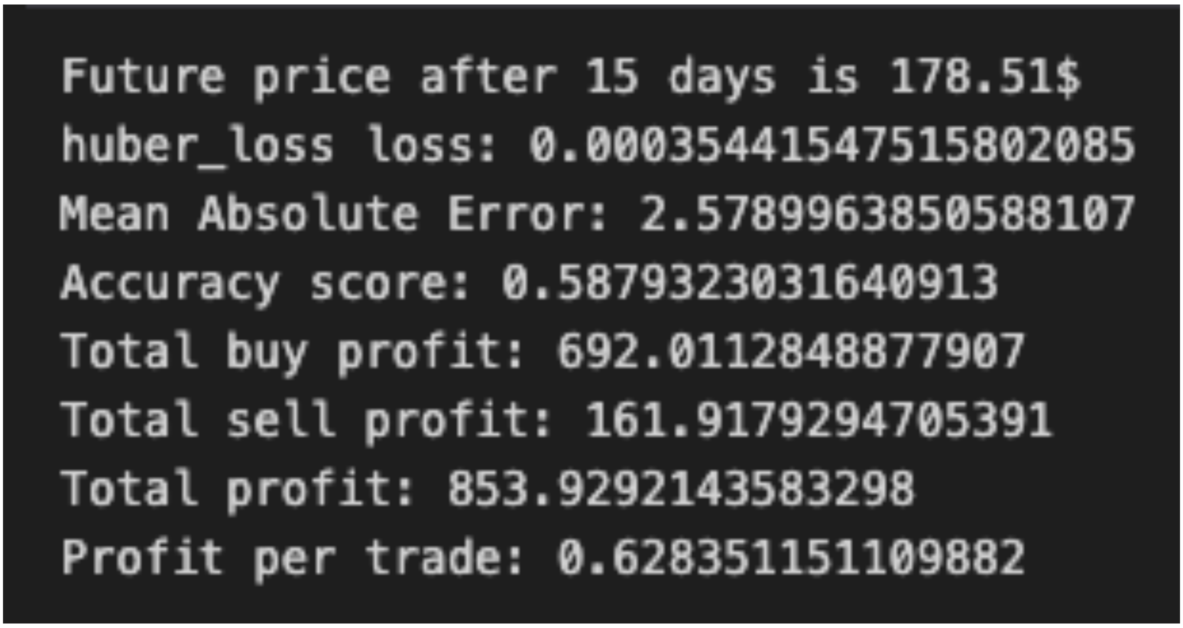
Also if run test.py successfully, we got the console print with details (the code print method demonstrates that at its python file)

2. Running code at Jupyter Notebook (.ipynb file) at Visual Studio Code



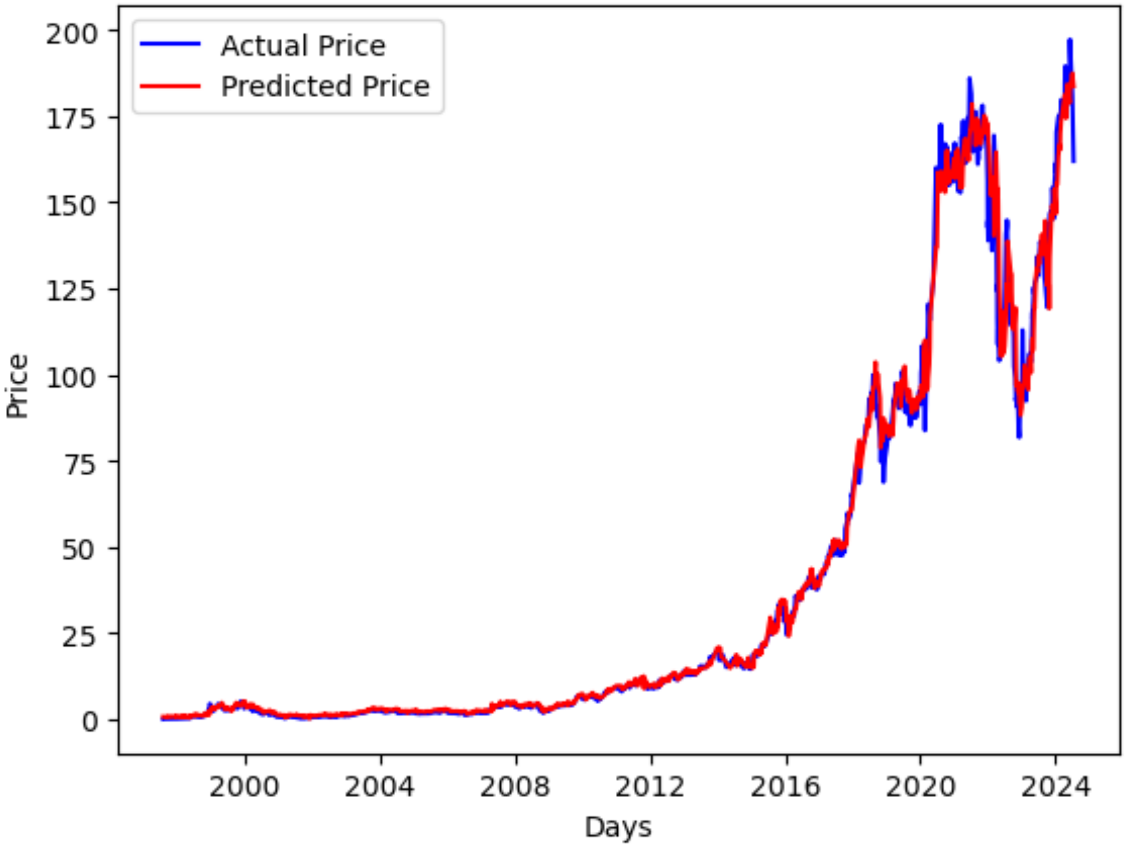
```
[10]   ✓  264m 59.6s

...    Epoch 1/500
       2024-08-14 15:14:08.9
       2024-08-14 15:14:09.0
```

For .ipynb (Jupyter Notebook) file, it takes around 2.5 hours to train the model (epoch 500)

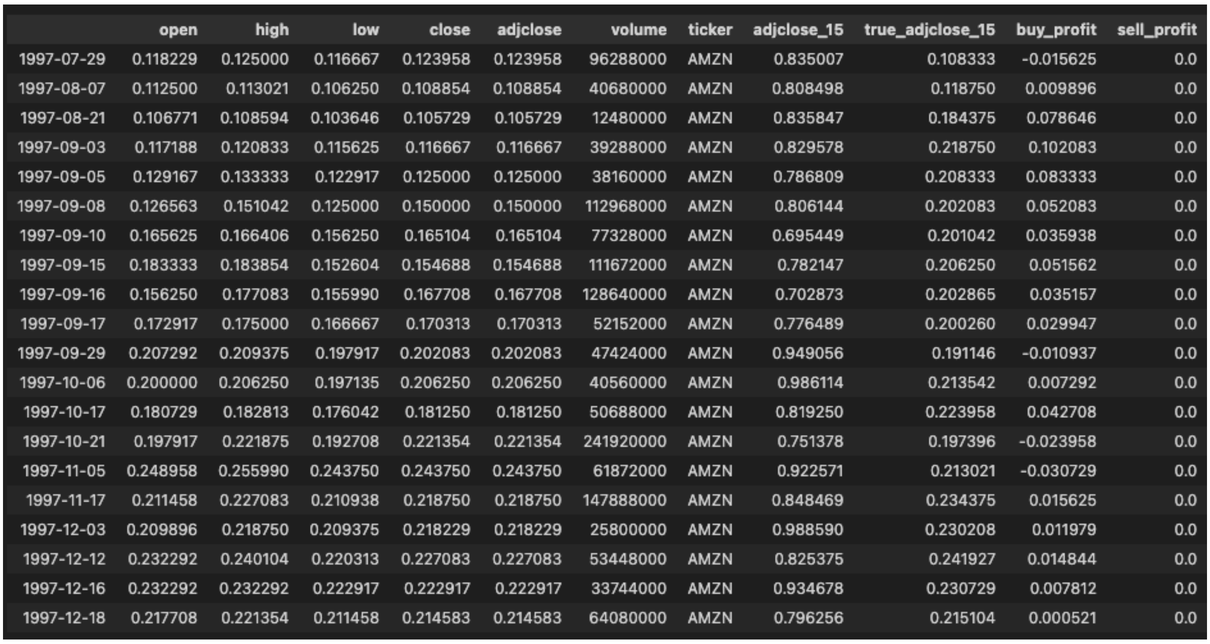


```
Future price after 15 days is 178.51$
huber_loss loss: 0.00035441547515802085
Mean Absolute Error: 2.5789963850588107
Accuracy score: 0.5879323031640913
Total buy profit: 692.0112848877907
Total sell profit: 161.9179294705391
Total profit: 853.9292143583298
Profit per trade: 0.628351151109882
```

The result of predicting the stock price (print method) at .ipynb file

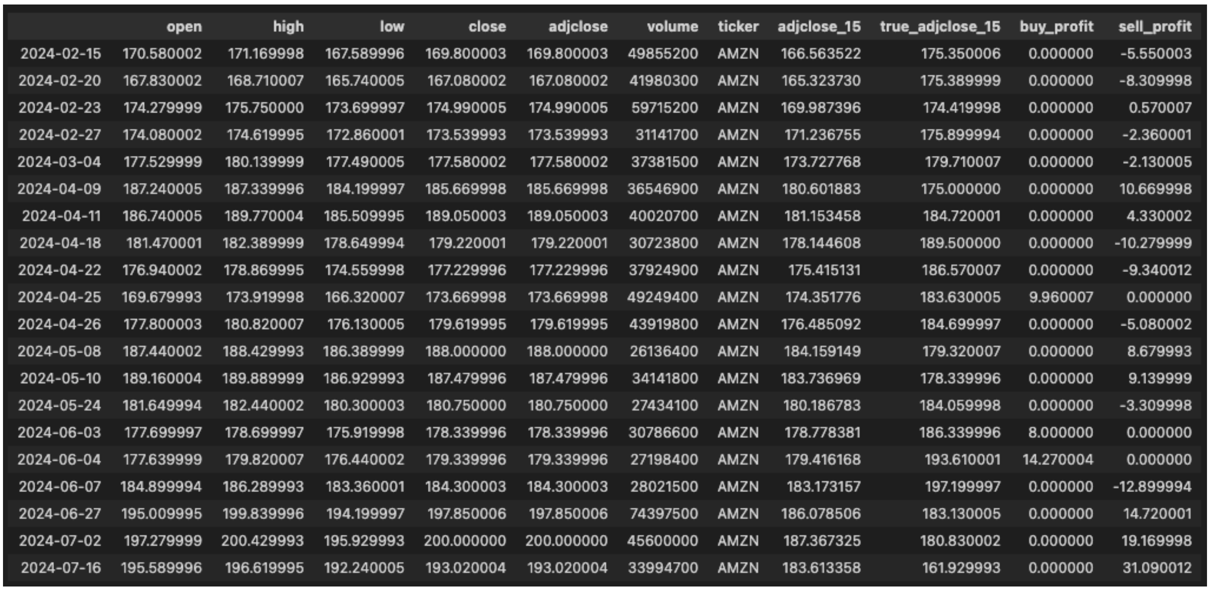


The plot graph demonstrate actual and predicted prices at jupyter notebook



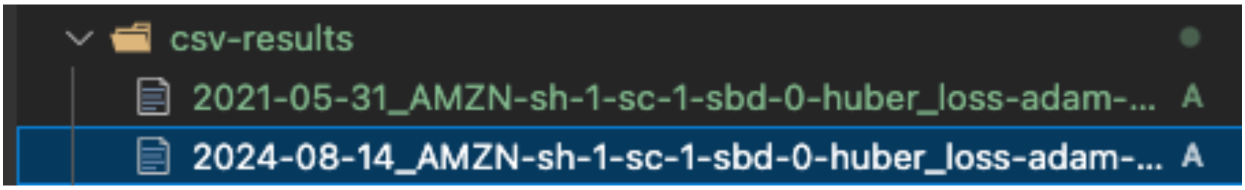
	open	high	low	close	adjclose	volume	ticker	adjclose_15	true_adjclose_15	buy_profit	sell_profit
1997-07-29	0.118229	0.125000	0.116667	0.123958	0.123958	96288000	AMZN	0.835007	0.108333	-0.015625	0.0
1997-08-07	0.112500	0.113021	0.106250	0.108854	0.108854	40680000	AMZN	0.808498	0.118750	0.009896	0.0
1997-08-21	0.106771	0.108594	0.103646	0.105729	0.105729	12480000	AMZN	0.835847	0.184375	0.078646	0.0
1997-09-03	0.117188	0.120833	0.115625	0.116667	0.116667	39288000	AMZN	0.829578	0.218750	0.102083	0.0
1997-09-05	0.129167	0.133333	0.122917	0.125000	0.125000	38160000	AMZN	0.786809	0.208333	0.083333	0.0
1997-09-08	0.126563	0.151042	0.125000	0.150000	0.150000	112968000	AMZN	0.806144	0.202083	0.052083	0.0
1997-09-10	0.165625	0.166406	0.156250	0.165104	0.165104	77328000	AMZN	0.695449	0.201042	0.035938	0.0
1997-09-15	0.183333	0.183854	0.152604	0.154688	0.154688	111672000	AMZN	0.782147	0.206250	0.051562	0.0
1997-09-16	0.156250	0.177083	0.155990	0.167708	0.167708	128640000	AMZN	0.702873	0.202865	0.035157	0.0
1997-09-17	0.172917	0.175000	0.166667	0.170313	0.170313	52152000	AMZN	0.776489	0.200260	0.029947	0.0
1997-09-29	0.207292	0.209375	0.197917	0.202083	0.202083	47424000	AMZN	0.949056	0.191146	-0.010937	0.0
1997-10-06	0.200000	0.206250	0.197135	0.206250	0.206250	40560000	AMZN	0.986114	0.213542	0.007292	0.0
1997-10-17	0.180729	0.182813	0.176042	0.181250	0.181250	50688000	AMZN	0.819250	0.223958	0.042708	0.0
1997-10-21	0.197917	0.221875	0.192708	0.221354	0.221354	241920000	AMZN	0.751378	0.197396	-0.023958	0.0
1997-11-05	0.248958	0.255990	0.243750	0.243750	0.243750	61872000	AMZN	0.922571	0.213021	-0.030729	0.0
1997-11-17	0.211458	0.227083	0.210938	0.218750	0.218750	147888000	AMZN	0.848469	0.234375	0.015625	0.0
1997-12-03	0.209896	0.218750	0.209375	0.218229	0.218229	25800000	AMZN	0.988590	0.230208	0.011979	0.0
1997-12-12	0.232292	0.240104	0.220313	0.227083	0.227083	53448000	AMZN	0.825375	0.241927	0.014844	0.0
1997-12-16	0.232292	0.232292	0.222917	0.222917	0.222917	33744000	AMZN	0.934678	0.230729	0.007812	0.0
1997-12-18	0.217708	0.221354	0.211458	0.214583	0.214583	64080000	AMZN	0.796256	0.215104	0.000521	0.0

Display 20 first rows at juputer notebook .ipynb



	open	high	low	close	adjclose	volume	ticker	adjclose_15	true_adjclose_15	buy_profit	sell_profit
2024-02-15	170.580002	171.169998	167.589996	169.800003	169.800003	49855200	AMZN	166.563522	175.350006	0.000000	-5.550003
2024-02-20	167.830002	168.710007	165.740005	167.080002	167.080002	41980300	AMZN	165.323730	175.389999	0.000000	-8.309998
2024-02-23	174.279999	175.750000	173.699997	174.990005	174.990005	59715200	AMZN	169.987396	174.419998	0.000000	0.570007
2024-02-27	174.080002	174.619995	172.860001	173.539993	173.539993	31141700	AMZN	171.236755	175.899994	0.000000	-2.360001
2024-03-04	177.529999	180.139999	177.490005	177.580002	177.580002	37381500	AMZN	173.727768	179.710007	0.000000	-2.130005
2024-04-09	187.240005	187.339996	184.199997	185.669998	185.669998	36546900	AMZN	180.601883	175.000000	0.000000	10.669998
2024-04-11	186.740005	189.770004	185.509995	189.050003	189.050003	40020700	AMZN	181.153458	184.720001	0.000000	4.330002
2024-04-18	181.470001	182.389999	178.649994	179.220001	179.220001	30723800	AMZN	178.144608	189.500000	0.000000	-10.279999
2024-04-22	176.940002	178.869995	174.559998	177.229996	177.229996	37924900	AMZN	175.415131	186.570007	0.000000	-9.340012
2024-04-25	169.679993	173.919998	166.320007	173.669998	173.669998	49249400	AMZN	174.351776	183.630005	9.960007	0.000000
2024-04-26	177.800003	180.820007	176.130005	179.619995	179.619995	43919800	AMZN	176.485092	184.699997	0.000000	-5.080002
2024-05-08	187.440002	188.429993	186.389999	188.000000	188.000000	26136400	AMZN	184.159149	179.320007	0.000000	8.679993
2024-05-10	189.160004	189.889999	186.929993	187.479996	187.479996	34141800	AMZN	183.736969	178.339996	0.000000	9.139999
2024-05-24	181.649994	182.440002	180.300003	180.750000	180.750000	27434100	AMZN	180.186783	184.059998	0.000000	-3.309998
2024-06-03	177.699997	178.699997	175.919998	178.339996	178.339996	30786600	AMZN	178.778381	186.339996	8.000000	0.000000
2024-06-04	177.639999	179.820007	176.440002	179.339996	179.339996	27198400	AMZN	179.416168	193.610001	14.270004	0.000000
2024-06-07	184.899994	186.289993	183.360001	184.300003	184.300003	28021500	AMZN	183.173157	197.199997	0.000000	-12.899994
2024-06-27	195.009995	199.839996	194.199997	197.850006	197.850006	74397500	AMZN	186.078506	183.130005	0.000000	14.720001
2024-07-02	197.279999	200.429993	195.929993	200.000000	200.000000	45600000	AMZN	187.367325	180.830002	0.000000	19.169998
2024-07-16	195.589996	196.619995	192.240005	193.020004	193.020004	33994700	AMZN	183.613358	161.929993	0.000000	31.090012

Display 20 last rows



csv-results

2021-05-31_AMZN-sh-1-sc-1-sbd-0-huber_loss-adam-... A

2024-08-14_AMZN-sh-1-sc-1-sbd-0-huber_loss-adam-... A

New csv file after export (to_csv method) at jupyter notebook .ipynb

Understanding of initial code base v0.1

This code base is about using the existing data from the Meta company (previously called Facebook) to predict stock price based on historical data. By using processing data on the specific datetime, using Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM).
Key function:

- Step 1 – Use required libraries, especially yfinance for financial data API from Yahoo Finance.
- Step 2 - Load data: choose the ticket symbol company (facebook like fb) and the timestamp that we want, two variables, which shows the start and end days, respectively. Then download method means find the company with ticket symbol with start and end date.
- Step 3 - Prepare data: Press all value into the value between 0 and 1 for neural network training. Predict the close price at scaled data in 60 days and add value to the value of x for time and y for price.
- Step 4 – Build the models: Use Sequential method to build neural network. Then include the layers of LSTM followed by Dropout to reduce overfitting in the model. The model add Dense layer to predict the next price. We try to compile the model in suitable optimizer and loss function, before fit for train the model at suitable epoch with a preferred batch size.
- Step 5 – Load test data: Define the date range for loading test data, before let them into a new variable for fetching the test data. After that, we try to combine both training and test data into the total variable, then try to predict based on the data to evaluate how accurate the model performs.
- Step 6 – Prediction + Plot: do the same way as training data, then scale them to back to the actual predicted prices.
- Step 7 – Visualisation: Show the plots between actual vs predicted prices, with labels and title demonstrates to make the plot graph looks easy and clear.
- Step 8 – Make the following day prediction: make the real data add to the following day, before make the prediction to scale back to the real value.