Intelligent Systems(COS 30018)

Vivek Saini(10382806)

Task B.1-Setup

Tutor: Mukesh Malani

In the First task that was given to us, we are presented with a model of stock prediction python code, which is used to predict the stock prices based on actual information from yahoo finance. Furthermore, we are required to install all the dependencies and libraries to run the code.

Youtube tutorial:

In the tutorial given to us, the person very well explains how the code is supposed to work, how to install the dependencies and what are the different blocks of the code are supposed to do. He also very well explains how the system is scaled and also the python program is required to make a graph and predict the stock prices based on the information in a certain time frame using the functions like:

```
# start = '2012-01-01', end='2017-01-01'
TRAIN_START = '2015-01-01'
TRAIN_END = '2020-01-01'

data =  yf.download(COMPANY, start=TRAIN_START, end=TRAIN_END, progress=False)
# yf.download(COMPANY, start = TRAIN_START, end=TRAIN_END)
```

<u>Observations:</u>

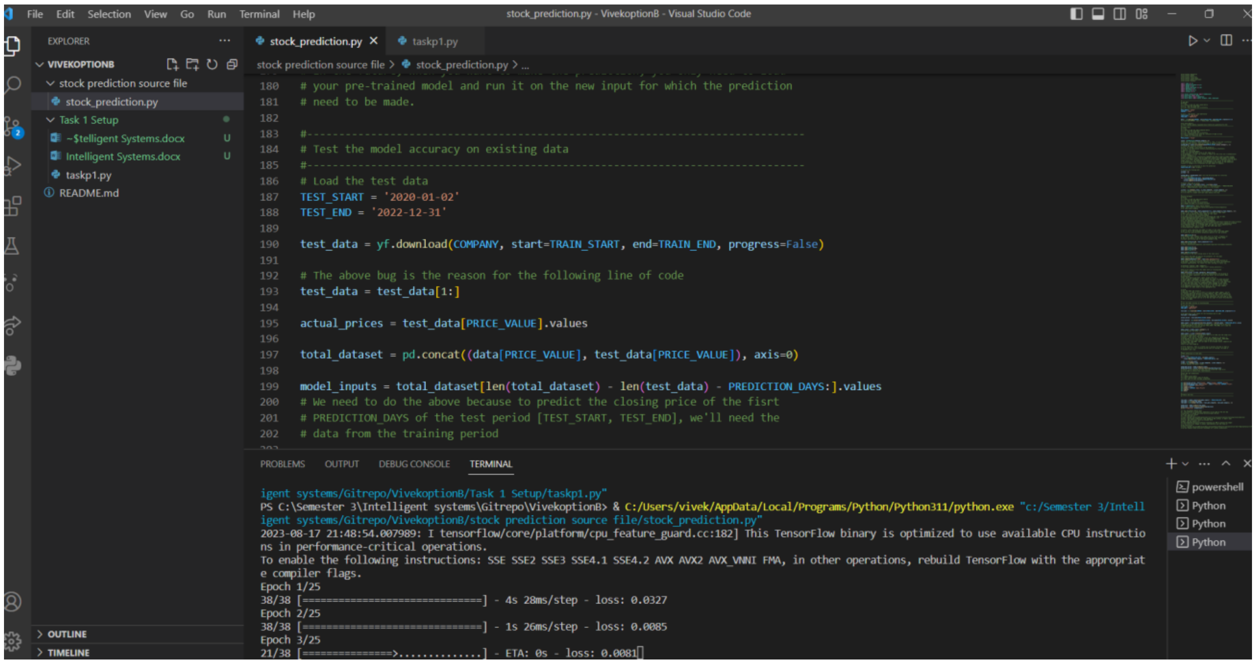
The observations that I made are as follows:

- There are different libraries that are required to be installed to collect the data and train the neural network model to produce the results as required.

- While testing the code, I encountered a problem where my keras model would not work, but I think there was some issue with the global version , I had to rework add few more lines to make it work , Here are the corrected libraries.

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

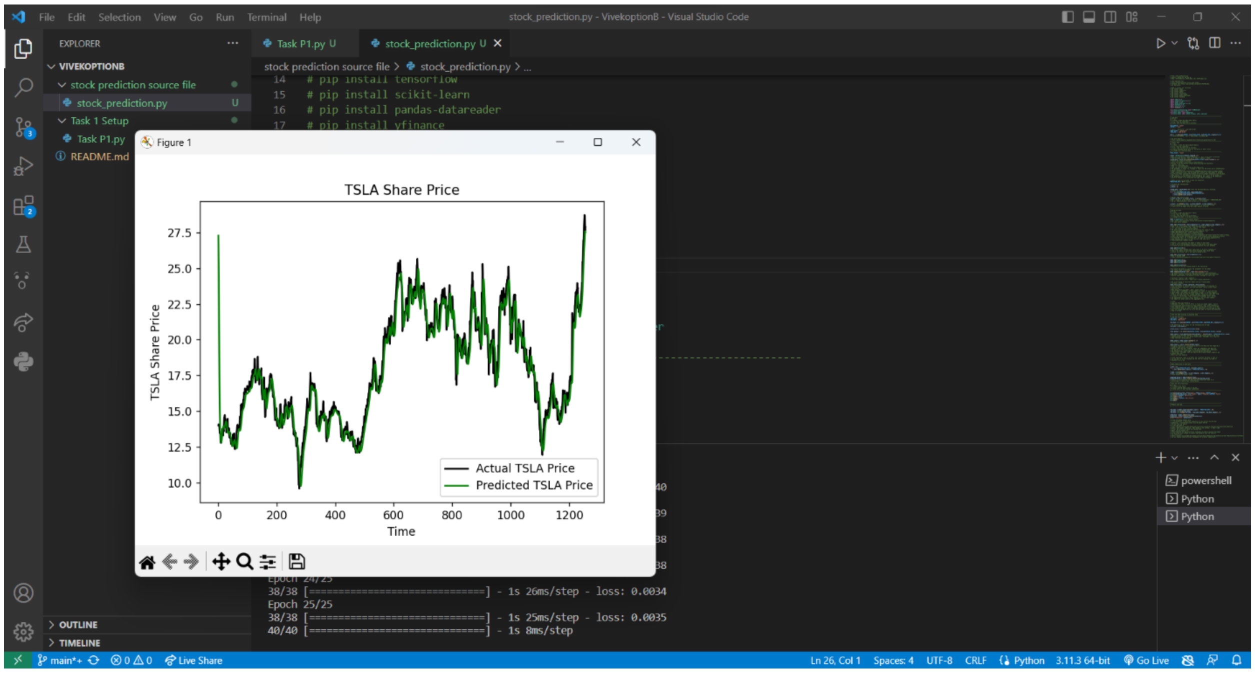
- Other than this I also observed that the model trains itself as explained in the tutorial and finds the most optimal route of predicting the stocks.


File Edit Selection View Go Run Terminal Help
stock_prediction.py - VivekoptionB - Visual Studio Code
EXPLORER
VIVEKOPTIONB
stock prediction source file
stock_prediction.py
Task 1 Setup
~$telligent Systems.docx
Intelligent Systems.docx
taskp1.py
README.md
OUTLINE
TIMELINE
stock_prediction.py
taskp1.py
stock prediction source file > stock_prediction.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell
Python
Python
Python

```
# your pre-trained model and run it on the new input for which the prediction
# need to be made.

#------------------------------------------------------------------------------
# Test the model accuracy on existing data
#------------------------------------------------------------------------------
# Load the test data
TEST_START = '2020-01-02'
TEST_END = '2022-12-31'

test_data = yf.download(COMPANY, start=TRAIN_START, end=TRAIN_END, progress=False)

# The above bug is the reason for the following line of code
test_data = test_data[1:]

actual_prices = test_data[PRICE_VALUE].values

total_dataset = pd.concat((data[PRICE_VALUE], test_data[PRICE_VALUE]), axis=0)

model_inputs = total_dataset[len(total_dataset) - len(test_data) - PREDICTION_DAYS:].values
# We need to do the above because to predict the closing price of the fisrt
# PREDICTION_DAYS of the test period [TEST_START, TEST_END], we'll need the
# data from the training period
```

```
igent systems/Gitrepo/VivekoptionB/Task 1 Setup/taskp1.py"
PS C:\Semester 3\Intelligent systems\Gitrepo\VivekoptionB> & C:/Users/vivek/AppData/Local/Programs/Python/Python311/python.exe "c:/Semester 3/Intell
igent systems/Gitrepo/VivekoptionB/stock prediction source file/stock_prediction.py"
2023-08-17 21:48:54.007989: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2 AVX AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriat
e compiler flags.
Epoch 1/25
38/38 [==============================] - 4s 28ms/step - loss: 0.0327
Epoch 2/25
38/38 [==============================] - 1s 26ms/step - loss: 0.0085
Epoch 3/25
21/38 [===============>..............] - ETA: 0s - loss: 0.0081
```

- The predicted prices were really similar to the actual prices.



```
# pip install tensorflow
# pip install scikit-learn
# pip install pandas-datareader
# pip install yfinance
```

```
Epoch 24/25
38/38 [==============================] - 1s 26ms/step - loss: 0.0034
Epoch 25/25
38/38 [==============================] - 1s 25ms/step - loss: 0.0035
40/40 [==============================] - 1s 8ms/step
```

Task P1:

This task was a little challenging, I got a lot of errors while trying to to set up the environment but in the end, it was fun to play with , It was similar to the first task, we installed the libraries, loaded them up, and wrote down the code which preprocesses the information.

After this we move on to the training of the model. So, we're setting up a neural network to predict future prices, and here's how we're going about it. First, we've got this thing called TEST_SIZE, which is basically the chunk of data we'll use for testing, and in this case, it's 20% of the whole dataset. Now, for the actual prediction part, we're picking out certain features from the data, and we call these FEATURE_COLUMNS.

Next up, we're talking about the actual structure of the neural network. We've got these layers, and you can choose how many you want with N_LAYERS. Inside these layers, we're using these cells, and by default, they're LSTM cells, which are good for sequences of data like this. You can decide how many cell units you want in each layer with UNITS.

But hey, we don't want the network to get too good at memorizing stuff and not really understanding, so we're adding this thing called DROPOUT. It's like telling some of the nodes in the layers to chill out during training – a dropout rate of 0.0 means no chill, while higher values mean more chill.

Now, BIDIRECTIONAL is a fancy term. It's about whether the network should consider the context from both past and future data points. So if it's set to true, it's like the network is reading the data in both directions.

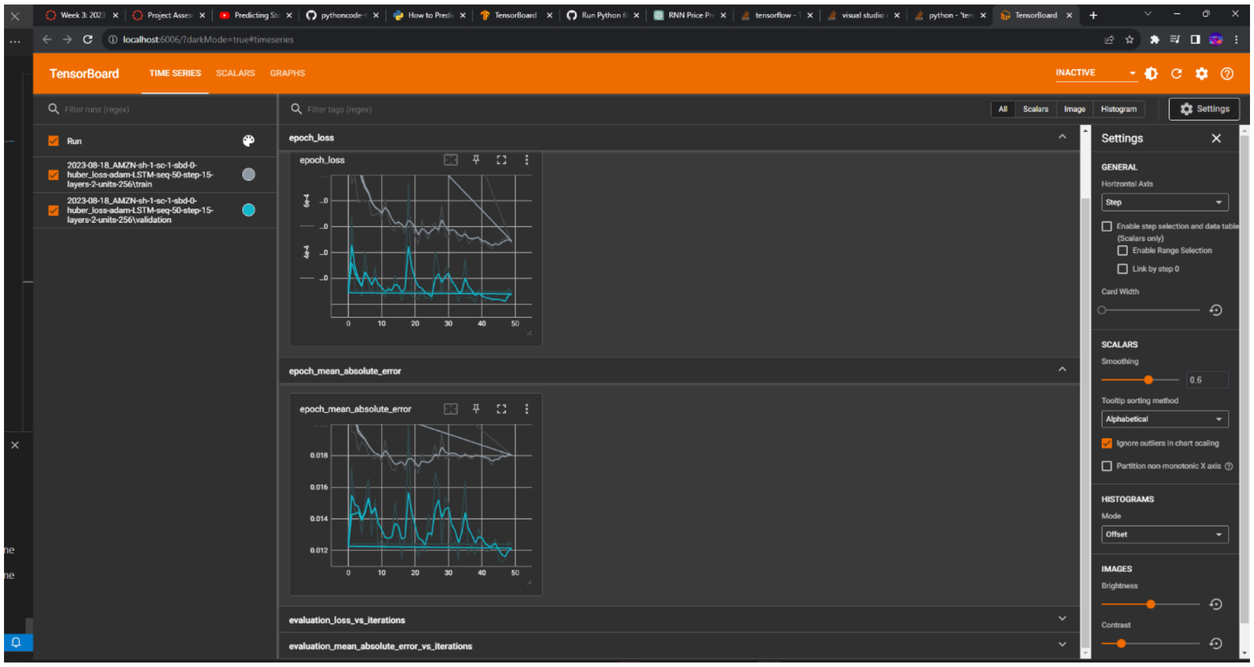
Okay, let's talk about the nitty-gritty of training. We've got this thing called LOSS, and we're going with Huber loss. It's a way to measure how wrong our predictions are, but it's not overly sensitive to those occasional big errors that can mess things up. Other options are mean absolute error (mae) and mean squared error (mse), which you might have heard of.

Then comes the OPTIMIZER, which is how the network tweaks itself to get better at predictions. We're defaulting to Adam here, which is a popular choice. During training, we don't want to feed the entire dataset at once – that's where BATCH_SIZE comes in. It's like serving up chunks of data for the network to learn from.

Lastly, we've got EPOCHS – it's the number of times the network goes through the entire training dataset. We're starting with 500 rounds, but you could try more to really let the network learn the patterns. So, in a nutshell, that's the setup we're going with to make our neural network predict those prices.

We also added the folders to make sure that results, logs , and data folders exist.

Then we moved on to check the progress by using the tensorboard command to check the mean squared or absolute error.



Run
2023-08-18_AMZN-sh-1-sc-1-sbd-0-huber_loss-adam-LSTM-seq-50-step-15-layers-2-units-256\train
2023-08-18_AMZN-sh-1-sc-1-sbd-0-huber_loss-adam-LSTM-seq-50-step-15-layers-2-units-256\validation

Settings

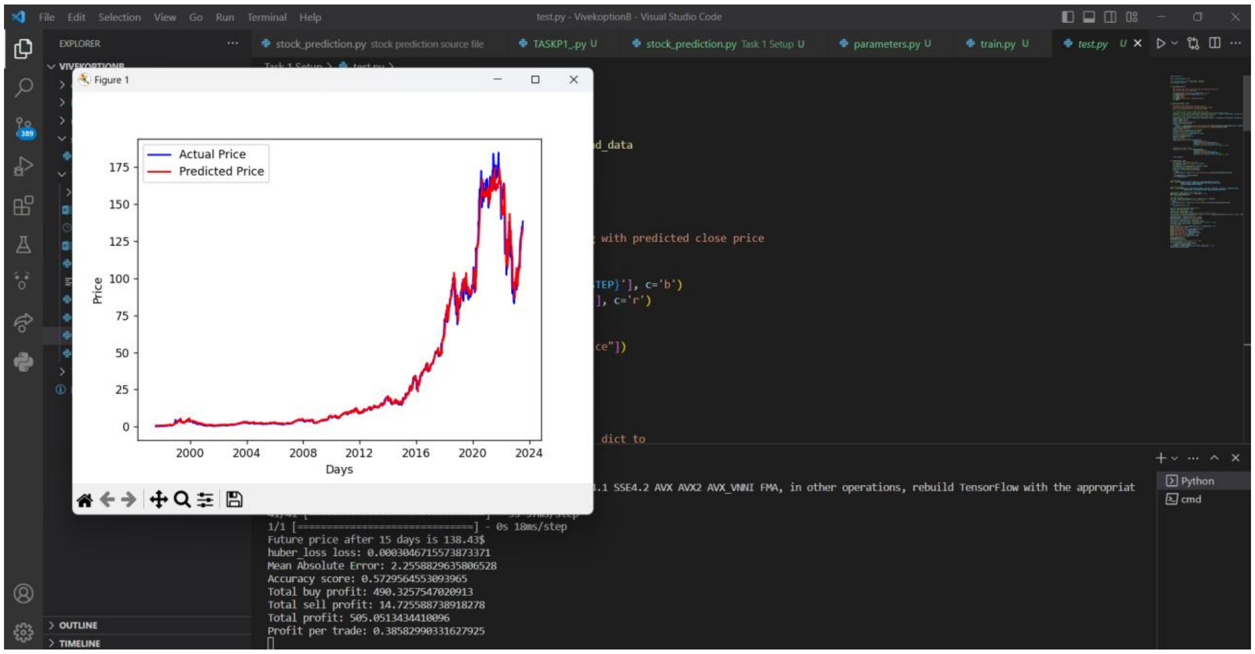
GENERAL
Horizontal Axis: Step
Enable step selection and data table (Scalars only)
Enable Range Selection
Link by step 0
Card Width

SCALARS
Smoothing: 0.6
Tooltip sorting method: Alphabetical
[✓] Ignore outliers in chart scaling
Partition non-monotonic X axis

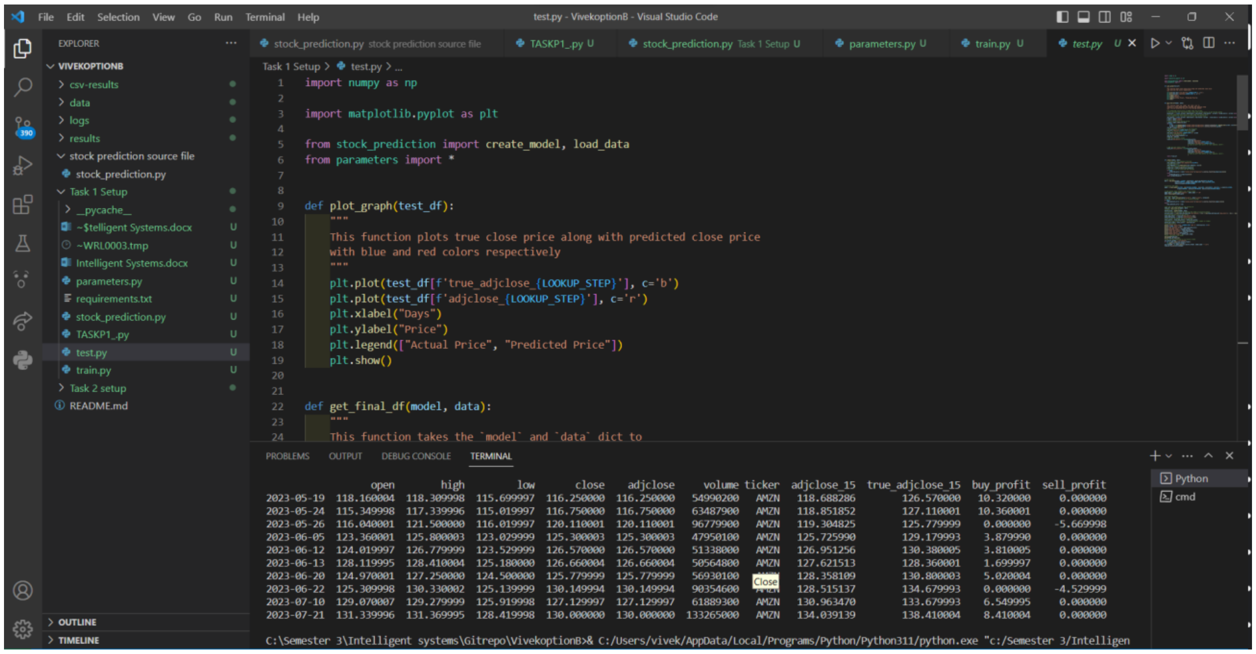
HISTOGRAMS
Mode: Offset

IMAGES
Brightness
Contrast

At last I would like to conclude this task with the final output after training the model.



```
.1 SSE4.2 AVX AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriat
1/1 [==============================] - 0s 18ms/step
Future price after 15 days is 138.43$
huber_loss loss: 0.0003046715573873371
Mean Absolute Error: 2.2558829635806528
Accuracy score: 0.5729564553093965
Total buy profit: 490.3257547020913
Total sell profit: 14.725588738918278
Total profit: 505.0513434410096
Profit per trade: 0.38582990331627925
```



```python
import numpy as np

import matplotlib.pyplot as plt

from stock_prediction import create_model, load_data
from parameters import *


def plot_graph(test_df):
    """
    This function plots true close price along with predicted close price
    with blue and red colors respectively
    """
    plt.plot(test_df[f'true_adjclose_{LOOKUP_STEP}'], c='b')
    plt.plot(test_df[f'adjclose_{LOOKUP_STEP}'], c='r')
    plt.xlabel("Days")
    plt.ylabel("Price")
    plt.legend(["Actual Price", "Predicted Price"])
    plt.show()


def get_final_df(model, data):
    """
    This function takes the `model` and `data` dict to
```

```
                  open        high         low       close    adjclose     volume ticker  adjclose_15  true_adjclose_15  buy_profit  sell_profit
2023-05-19  118.160004  118.309998  115.699997  116.250000  116.250000   54990200   AMZN   118.688286        126.570000   10.320000     0.000000
2023-05-24  115.349998  117.339996  115.019997  116.750000  116.750000   63487900   AMZN   118.851852        127.110001   10.360001     0.000000
2023-05-26  116.040001  121.500000  116.019997  120.110001  120.110001   96779900   AMZN   119.304825        125.779999    0.000000    -5.669998
2023-06-05  123.360001  125.800003  123.029999  125.300003  125.300003   47950100   AMZN   125.725990        129.179993    3.879990     0.000000
2023-06-12  124.019997  126.779999  123.529999  126.570000  126.570000   51338000   AMZN   126.951256        130.380005    3.810005     0.000000
2023-06-13  128.119995  128.410004  125.180000  126.660004  126.660004   50564800   AMZN   127.621513        128.360001    1.699997     0.000000
2023-06-20  124.970001  127.250000  124.500000  125.779999  125.779999   56930100   Close  128.358109        130.800003    5.020004     0.000000
2023-06-22  125.309998  130.330002  125.139999  130.149994  130.149994   90354600   AMZN   128.515137        134.679993    0.000000    -4.529999
2023-07-10  129.070007  129.279999  125.919998  127.129997  127.129997   61889300   AMZN   130.963470        133.679993    6.549995     0.000000
2023-07-21  131.339996  131.369995  128.419998  130.000000  130.000000  133265000   AMZN   134.039139        138.410004    8.410004     0.000000

C:\Semester 3\Intelligent systems\Gitrepo\VivekoptionB>& C:/Users/vivek/AppData/Local/Programs/Python/Python311/python.exe "c:/Semester 3/Intelligen
```

Task P2:

I tried running the code in a virtual environment but was unsuccessful in running it, although I found the youtube tutorial for that code and had a good understanding of how it works, to summarize it is similar to the code I have done before. In this scenario we have different libraries to install but similar practice to load the data, train the data , and then testing out the data. In the end, we make a candlestick chart instead of a line graph which we did with matplotlib.