MSc Business Analytics

MIS41110 Programming for Analytics

Report

UCD Smurfit School of Business
Student Name: Vaishnavi Milind Gadekar
Student ID: 20200073

Stock Analysis Assignment

Vaishnavi's Stock Application Provides a graphical user interface with the following multiple menu options:

1. Search stocks and perform descriptive analytics on the data.
2. Query time range
3. Data Visualization
4. Stock Market Prediction using Linear Regression

The main menu contains 6 options, and whenever we run the StockQuotes.py, we get the menu and the need to input our choice. The code will then call the functions according to the user input.

The main menu after running StockQuotes.py:

```
In [65]: runfile('C:/Users/hp/Desktop/PythonProject/Code_Vaishnavi/StockQuotes.py',
wdir='C:/Users/hp/Desktop/PythonProject/Code_Vaishnavi')
Reloaded modules: StockPredictorLogic, DataVisualizationLogic,
DataVisualizationMenu
Welcome to the main menu of Vaishnavi's Stock Application!
The following are your options:
1. Search Stocks
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Main Menu
Please choose an option (0 to exit):
```

IPython console History

LSP Python: ready conda: base (Python 3.8.3) Line 117, Col 24 UTF-8 CRLF RW Mem 85%

1. Search Stocks:

This functionality allows the user to search the stocks of the Ticker Symbol provided by the user. The application finds that symbol in the companylist.csv in the same folder as the code.

The data is not downloaded or fetched live, but it is being read from a CSV file on the hard-drive.

```
1. Search Stocks
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): 1
Search Stocks

Please choose ticker symbol: AMZN
    Symbol              Name  ...                        Summary Quote Unnamed: 8
194   AMZN  Amazon.com, Inc.  ...  https://old.nasdaq.com/symbol/amzn        NaN

[1 rows x 9 columns]
       LastSale  IPOyear  Unnamed: 8
count      1.00      1.0         0.0
mean    3117.02   1997.0         NaN
std         NaN      NaN         NaN
min     3117.02   1997.0         NaN
25%     3117.02   1997.0         NaN
50%     3117.02   1997.0         NaN
75%     3117.02   1997.0         NaN
max     3117.02   1997.0         NaN

Main Menu
```

IPython console History

LSP Python: ready conda: base (Python 3.8.3) Line 146, Col 39 UTF-8 CRLF RW Mem 86%

2. Query Time Range:

- The application provides functionality to fetch the data for a particular time range and ticker Symbol.
- The data is being fetched live from Yahoo Finance using the Ticker() method by passing the ticker Symbol in the method.
- But we are not exporting the data into a CSV by downloading it from Yahoo Finance live, rather this is just fetching the data from Yahoo Finance and printing the data and the descriptive analytics performed on the data over UI.
- We have an excellent functionality to download the data by giving a data range, ticker symbol and exporting the data into a CSV file. The option 5 in the main menu will be exporting the data.
- This functionality is provided to the users when they don't want the file to be downloaded, rather just to print on the UI. Exporting the data is the next of this functionality.
- Yet, for flexibility, the application is providing both the options.
- Likewise, the application is also providing the option to search the stocks on Yahoo Finance in Real Time using YahooFinance library of Python and Pandas, numpy to process on the data fetched. Yet, we also have the functionality of searching the stocks from a CSV file, for flexibility of the users.

```
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): 2

Please choose ticker symbol: AMZN

Please enter period (1d, 5d, 1mo, 3mo, 6mo, 1y, 2y, 5y, 10y, ytd, max): 6mo
yfinance.Ticker object <AMZN>
                  Open         High  ...  Dividends  Stock Splits
Date                                  ...
2020-06-05  2444.510010  2488.649902  ...          0             0
2020-06-08  2500.199951  2530.000000  ...          0             0
2020-06-09  2529.439941  2626.429932  ...          0             0
2020-06-10  2645.000000  2722.350098  ...          0             0
2020-06-11  2603.500000  2671.379883  ...          0             0
...                  ...          ...  ...        ...           ...
2020-11-30  3208.479980  3228.389893  ...          0             0
2020-12-01  3188.500000  3248.949951  ...          0             0
2020-12-02  3221.649902  3232.000000  ...          0             0
2020-12-03  3205.459961  3228.639893  ...          0             0
2020-12-04  3198.209961  3198.209961  ...          0             0

[128 rows x 7 columns]
```

IPython console History

LSP Python: ready | conda: base (Python 3.8.3) | Line 149, Col 31 | UTF-8 | CRLF | RW | Mem 85%

The application also has the check for invalid Ticker symbols, date ranges and durations: (Exception Handling)

- Invalid duration:

```
Main Menu
1. Search Stocks
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): 1

Please enter a ticker symbol: AMZN

Please enter period (1d, 5d, 1mo, 3mo, 6mo, 1y, 2y, 5y, 10y, ytd, max): 3d
Enter a valid duration period!

Please enter a ticker symbol:
```

IPython console History

LSP Python: ready | conda: base (Python 3.8.3) | Line 74, Col 26 | UTF-8 | CRLF | RW | Mem 86%

- The Application also checks for invalid Ticker Symbol

```
Main Menu
1. Search Stocks
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): 3

Welcome to the Data Visualization Sub-menu!

Let's take the parameters (Ticker Symbol, Start Date and End Date, so you can view
different type of plots for these specific parameters


Please choose ticker symbol: vaish

Please input the start date (YYYY-MM-DD): 2020-01-01

Please input the end date (YYYY-MM-DD): 2020-01-10
'regularMarketOpen'
Oops! You entered a invalid Ticker symbol. Please re-enter.

Please choose ticker symbol:
```

IPython console History

LSP Python: ready | conda: base (Python 3.8.3) | Line 96, Col 46 | UTF-8 | CRLF | RW | Mem 87%

- Invalid Date Range:

```
Please choose ticker symbol: vaish

Please input the start date (YYYY-MM-DD): 2020-01-01

Please input the end date (YYYY-MM-DD): 2020-01-10
'regularMarketOpen'
Oops! You entered a invalid Ticker symbol. Please re-enter.

Please choose ticker symbol: AMZN

Please input the start date (YYYY-MM-DD): 2020-02-01

Please input the end date (YYYY-MM-DD): 2020-01-01
End date value should be '>=' the Start date value.
 Kindly enter the correct values.

Please choose ticker symbol:
```

IPython console History

LSP Python: ready | conda: base (Python 3.8.3) | Line 96, Col 46 | UTF-8 | CRLF | RW | Mem 86%

3. Data Visualization:

In the main menu, 3rd option is for Data Visualization sub-menu. By clicking on 3, we get the following message on the UI:

```
In [65]: runfile('C:/Users/hp/Desktop/PythonProject/Code_Vaishnavi/StockQuotes.py',
wdir='C:/Users/hp/Desktop/PythonProject/Code_Vaishnavi')
Reloaded modules: StockPredictorLogic, DataVisualizationLogic,
DataVisualizationMenu
Welcome to the main menu of Vaishnavi's Stock Application!
The following are your options:
1. Search Stocks
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Main Menu
Please choose an option (0 to exit): 3

Welcome to the Data Visualization Sub-menu!

Let's take the parameters (Ticker Symbol, Start Date and End Date, so you can view
different type of plots for these specific parameters


Please choose ticker symbol:
```

IPython console History

LSP Python: ready conda: base (Python 3.8.3) Line 117, Col 24 UTF-8 CRLF RW Mem 86%

- The Numpy library is used for calculating the mean, moving averages, etc mathematical values, YahooFinance is used to fetch the data and process on it like getting the descriptive statistics of the stock data.
- Pandas is used to read from and write to files, processing on the Dataframes and the matplotlib library is used to plot the graphs.
- The usage of these libraries makes the implementation of the code easy.
- In this menu, the first thing the user is prompted is to enter the ticker symbol, start date and end-date, and using these parameters the application will be plotting the graphs.
- The user will see a message which will tell them they are in the Data Visualization menu and what all types of visualizations of data are available.

```
Welcome to the Data Visualization Sub-menu!

Let's take the parameters (Ticker Symbol, Start Date and End Date, so you can view
different type of plots for these specific parameters


Please choose ticker symbol: AMZN

Please input the start date (YYYY-MM-DD): 2020-01-01

Please input the end date (YYYY-MM-DD): 2020-12-06


Welcome to the Data visualization board.
The following are your choices:

1. Closing Price vs Time
2. Adjacent Closing Price vs Time
3. Open and Closing Prices vs Time
4. Moving Average
5. Weighted Moving Average
6. Moving Average Convergence/Divergence
7. Linear Trend Lines


Please enter your choice to create plots (8 to exit):
```

IPython console History

LSP Python: ready conda: base (Python 3.8.3) Line 83, Col 13 UTF-8 CRLF RW Mem 82%

The application provides with a sub-menu inside the Data Visualization menu, offering the following types of graphs and plots depicting stock trends and patterns:

1. Closing Price vs time plot
2. Adjacent Closing Price vs time plot
3. Open and Closing values vs time in the same graph for comparison
4. Moving Average
5. Weighted Moving Average
6. Moving Average Convergence/Divergence (MACD)
7. Linear Trend Lines

- Here, all the graphs would be built on the same parameters: Ticker Symbol, Start Date, End Date in one go. So, the user would not have to enter them each time, hence saving time and clicks.
- There are 3 graphs (Moving Average, Weighted Moving Average, Moving Average Convergence/Divergence) that need additional inputs from the user and the user would be prompted when he/she selects those options.
- When the user selects any option, 1 for Closing Price vs Time Plot, the user would be prompted asking if he/she wants to save the plot as a PNG file.

```
Welcome to the Data visualization board.
The following are your choices:

1. Closing Price vs Time
2. Adjacent Closing Price vs Time
3. Open and Closing Prices vs Time
4. Moving Average
5. Weighted Moving Average
6. Moving Average Convergence/Divergence
7. Linear Trend Lines


Please enter your choice to create plots (8 to exit): 1
[*********************100%***********************]  1 of 1 completed

Do you want to save the map in .png format?
If yes, please press Y else press N:
```

IPython console History

- If said yes (by pressing Y), the plot generated on the console would be saved in the directory where the code resides.
- After inputting Y, the plot will be saved, and would also be shown on the console along with a message informing the user that the plot is visible in the 'Plots' tab on console and that all the other graphs generated will also be using the same parameters.
- Additionally, if the user wants to generate the graphs for different parameters, the guide to that is given too. The application then prompts for the next input from the user (either plotting other graphs from the Data Visualization sub-menu or exiting and going back to the main menu)

```
Please enter your choice to create plots (8 to exit): 1
[*********************100%***********************]  1 of 1 completed

Do you want to save the map in .png format?
If yes, please press Y else press N: Y

The plot is visible in the 'Plots' tab

If you want to view technical indicator graphs for another Ticker Symbol and date
ramge, Enter 8 to exit the data visualization sub-menu, 0 to exit main menu, and
enter 3 to input your new parameters. Else, continue inputting other options to
plot graphs for existing parameters.

Please enter your choice to create plots (8 to exit):
```

IPython console History

LSP Python: ready | conda: base (Python 3.8.3) | Line 117, Col 24 | UTF-8 | CRLF | RW | Mem 87%

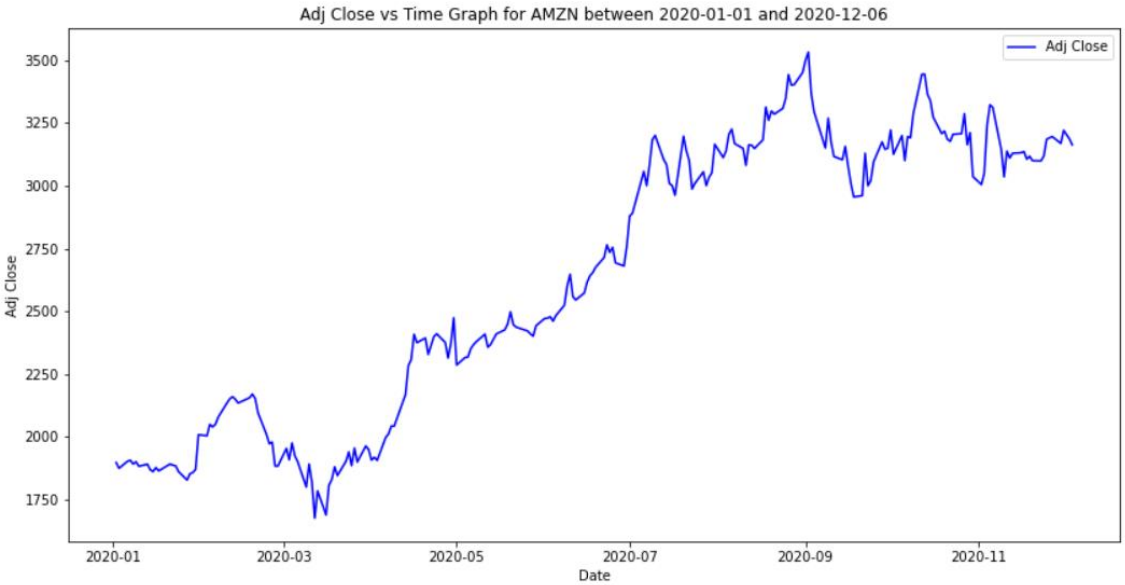
The parameters considered while plotting the following plots:

- Ticker Symbol: AMZN
- Start Date: 2020-01-01
- End Date: 2020-12-06

1. **Closing Price vs Time plot**

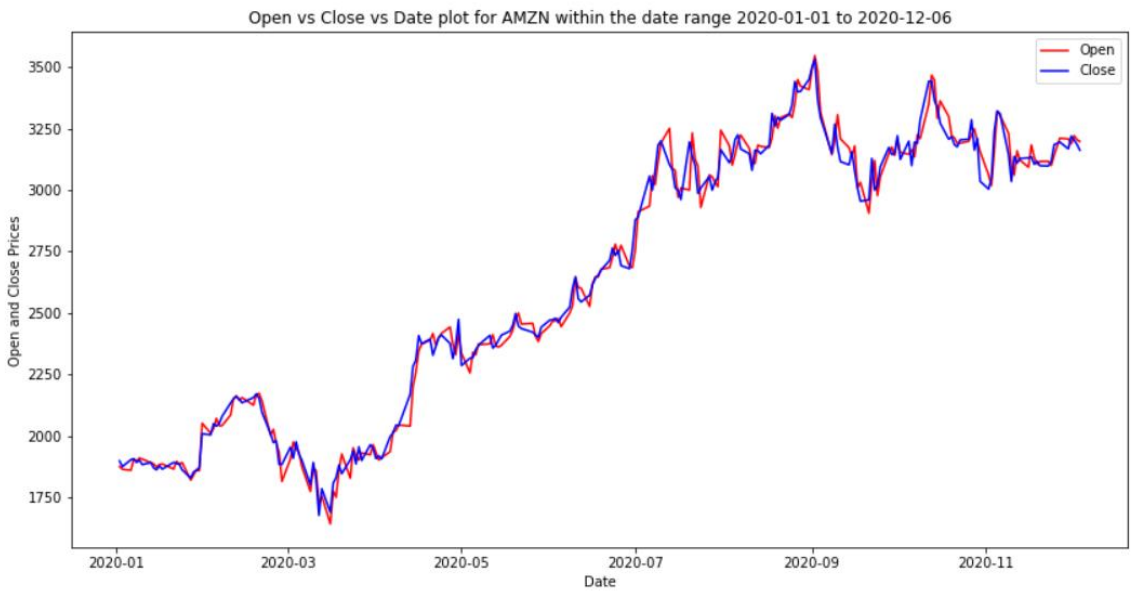
The above plot is of the Closing Price against Time. The graph is titled accordingly. The title is dynamic, which means that it would change the parameters as per the user's input.
The X-axis has the time and is labelled as Date. The Y-axis has Closing Price and is labelled the same. There is a legend on the top corner, depicting what the line means.

2. Adj Closing Price vs Time plot



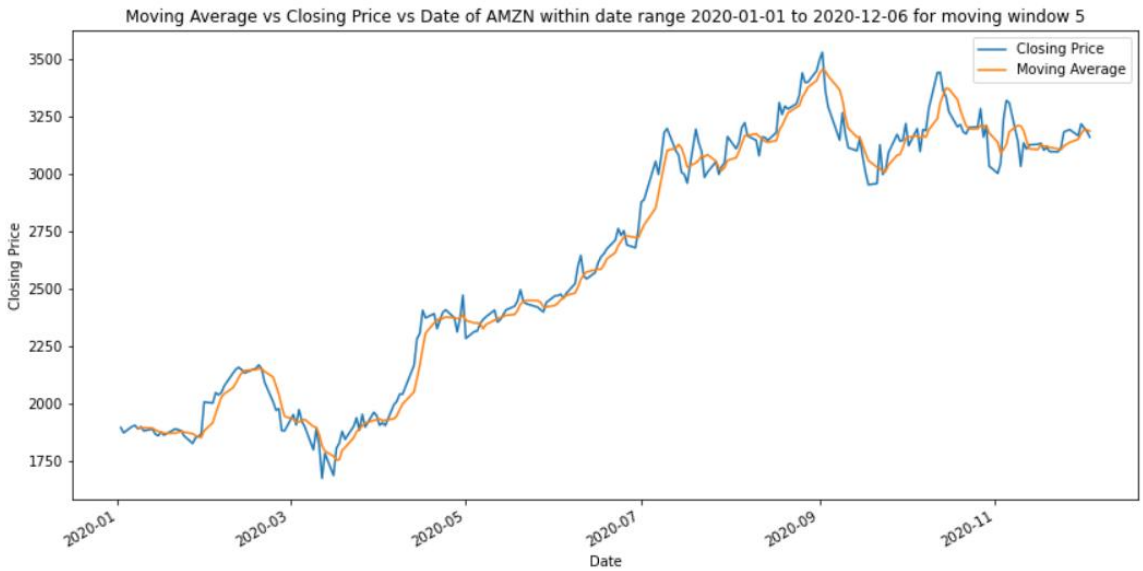
Similar to the above graph, time is plotted on X-axis, and Adj Closing Price on the Y-axis. The graph is titled as per the user's inputs (Ticker Symbol, Start Date, End Date) and the legend informs the user what color of the line graph represents what data.

3. Open and Close vs Time graph



This graph has 2 values being plotted simultaneously against time. This can help us in comparison of the Open and Closing values. Additionally, the legend tells that the red line represents the Open price trend of AMZN stock from 1st January 2020 to 6th December 2020. And blue line represents the Closing price for the same. The graph is titled, the axes are labelled appropriately.

4. Moving Average plot

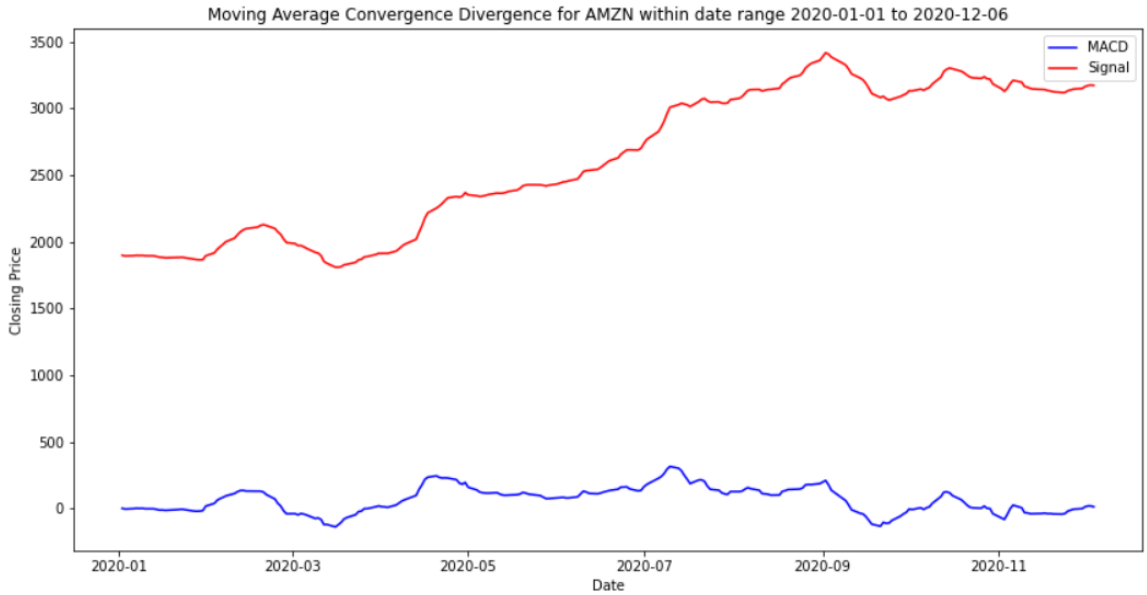


This plot has closing price and moving average plotted together. The legend depicts what color represents for what line and the axes and graph are labelled appropriately.

5. Weighted Moving Average

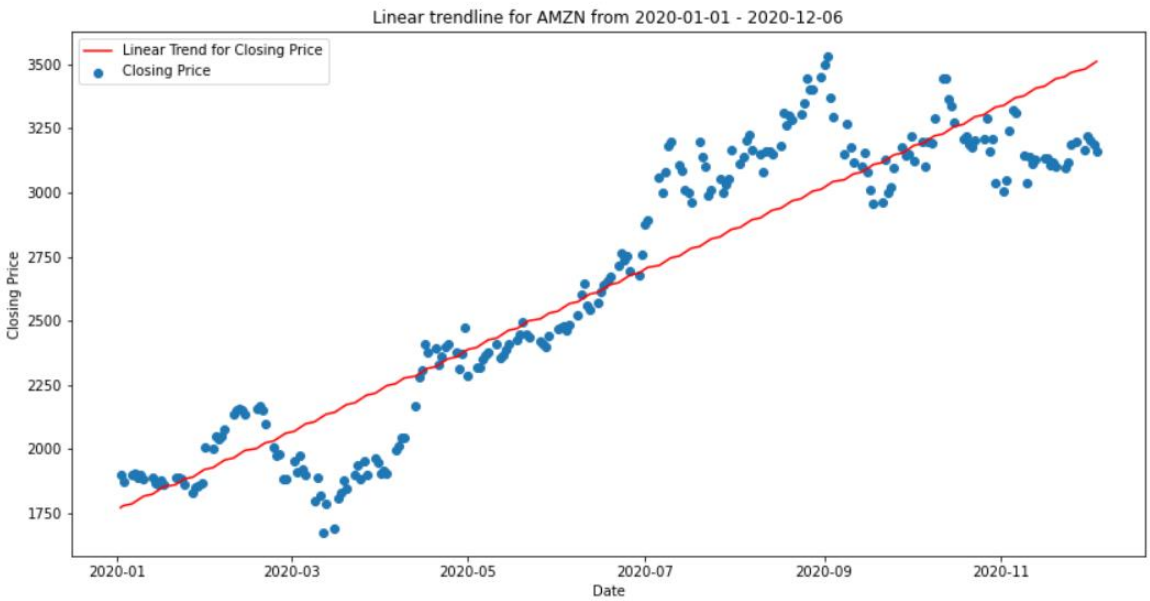
The X-axis has dates, and Y had closing prices. The plot has 3 things plotted, closing price, closing moving average and weighted moving average. Legend depicts what color line represents what value.

6. Moving Average Convergence/Divergence



The X-axis has date and Y has Closing Price of the stock data. The legend depicts what color line represents what value. The graph is titled appropriately as per the user's inputs.

7. Linear Trend Lines



This is a plot of closing price scatter and a linear trend of the closing price. The legend has tells the user what color represents what value. The axes and graph are labelled appropriately.

The application also provides the option to the user to change the parameters. The user is prompted after each plot and the guide is provided, with option numbers.

```
Do you want to save the map in .png format?
If yes, please press Y else press N: Y

The plot is visible in the 'Plots' tab

If you want to view technical indicator graphs for another Ticker Symbol and date
ramge, Enter 8 to exit the data visualization sub-menu, 0 to exit main menu, and
enter 3 to input your new parameters. Else, continue inputting other options to
plot graphs for existing parameters.

Please enter your choice to create plots (8 to exit): 8

Main Menu
Please choose an option (0 to exit): 3

Welcome to the Data Visualization Sub-menu!

Let's take the parameters (Ticker Symbol, Start Date and End Date, so you can view
different type of plots for these specific parameters


Please choose ticker symbol: NFLX

Please input the start date (YYYY-MM-DD): 2020-01-01

Please input the end date (YYYY-MM-DD): 2020-01-30
```

IPython console History

LSP Python: ready conda: base (Python 3.8.3) Line 117, Col 24 UTF-8 CRLF RW Mem 86%

Here, the user has changed the inputs to

Ticker Symbol: NFLX
Start Date: 2020-01-01
End Date: 2020-01-31

And the same code runs again on the same inputs, unless user changes the inputs again.

4. Stock Market Prediction

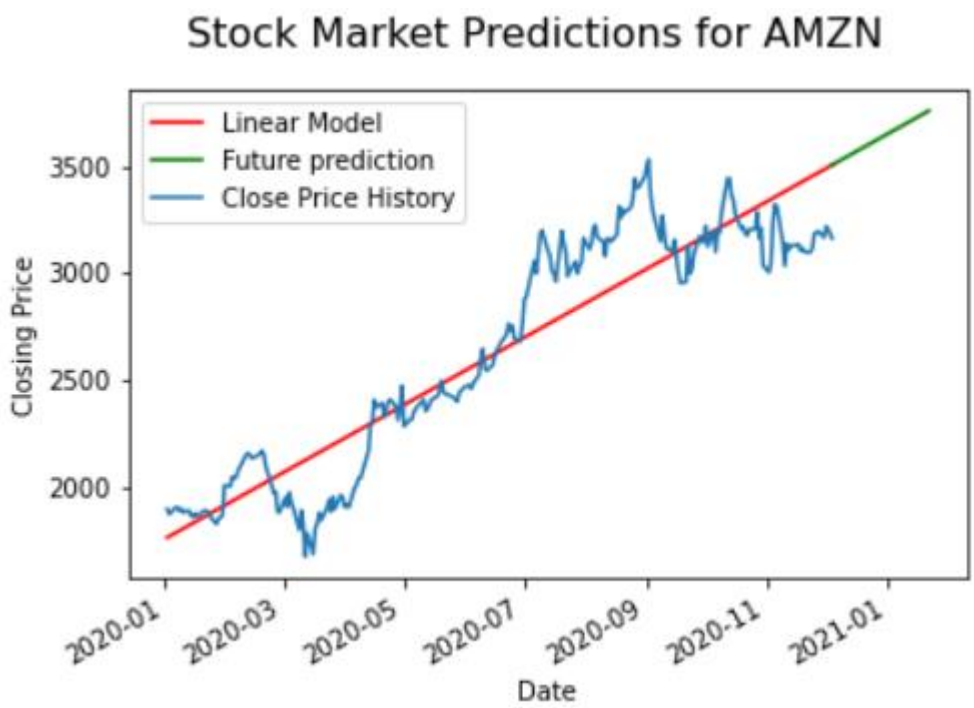
The Stock Market Prediction model uses Linear Regression to predict the values for the next n days which is the user input. 4th choice is the Stock Market Prediction and the application prompts the user to enter the Ticker symbol, the period for which the historical stock data is to be considered for modelling, and the number of days into the future the stock is to be predicted.

```
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): 4

Please choose ticker symbol: AMZN

Please enter period (1d, 5d, 1mo, 3mo, 6mo, 1y, 2y, 5y, 10y, ytd, max): ytd

Enter number of days for which you want to predict the closing price: 50

Do you want to save the map in .png format?
If yes, please press Y else press N: Y
Main Menu
1. Search Stocks
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit):
```

IPython console History

LSP Python: ready conda: base (Python 3.8.3) Line 123, Col 22 UTF-8 CRLF RW Mem 87%

The application uses the LinearRegression model to train the data and predict the values of the future dates.



The dates and predicted values are also printed out on the console

```
Please choose an option (0 to exit): 4

Please choose ticker symbol: AMZN

Please enter period (1d, 5d, 1mo, 3mo, 6mo, 1y, 2y, 5y, 10y, ytd, max): 5d

Enter number of days for which you want to predict the closing price: 5
Future Dates:
 DatetimeIndex(['2020-12-04', '2020-12-05', '2020-12-06', '2020-12-07',
               '2020-12-08'],
              dtype='datetime64[ns]', freq=None)
Stock Prices:
 [[3179.33803711]
 [3174.91103516]
 [3170.4840332 ]
 [3166.05703125]
 [3161.6300293 ]]

Do you want to save the map in .png format?
If yes, please press Y else press N:
```

IPython console History

LSP Python: ready conda: base (Python 3.8.3) Line 123, Col 22 UTF-8 CRLF RW Mem 85%

The application also provides the functionality to save the predicted values on specified number of days in the future. The data is saved in a CSV file. So that the users can just open the CSV and get the values directly.

__pycache__	06-12-2020 19:24	File folder	
AMZN_ADJC	06-12-2020 17:35	PNG File	38 KB
AMZN_CLOSE	06-12-2020 17:57	PNG File	38 KB
AMZN_LinearRegression	06-12-2020 19:28	PNG File	31 KB
AMZN_LTL	06-12-2020 17:55	PNG File	37 KB
AMZN_MA	06-12-2020 17:41	PNG File	56 KB
AMZN_MACD	06-12-2020 17:31	PNG File	35 KB
AMZN_OC	06-12-2020 17:37	PNG File	52 KB
AMZN_Prediction	06-12-2020 19:28	Microsoft Excel Co...	4 KB
AMZN_WMA	06-12-2020 17:55	PNG File	33 KB
companylist	20-11-2020 03:20	Microsoft Excel Co...	513 KB
DataVisualizationLogic.py	06-12-2020 18:08	PY File	7 KB
DataVisualizationMenu.py	06-12-2020 15:44	PY File	4 KB
file	06-12-2020 14:08	Microsoft Excel Co...	12 KB
NFLX_CLOSE	06-12-2020 18:05	PNG File	34 KB
StockPredictorLogic.py	06-12-2020 19:24	PY File	4 KB
StockQuotes.py	06-12-2020 18:10	PY File	6 KB
User Manual	06-12-2020 19:32	Microsoft Word D...	879 KB

The file is saved in the folder where the code resides, along with the other downloaded graphs.

5. Display Descriptive Analytics on console and Download Data to CSV file

Display Descriptive Statistics and Export the downloaded data in a CSV file in the folder where the code resides.

The user can provide the parameters: Ticker symbol, start date, end date and the application display the data and it's descriptive analytics on the console, and also provides the feature to export the downloaded data into a CSV file.

```
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): 5
Let's take parameters to download the data in CSV format along with descriptive
statistics.

Please choose ticker symbol: AMZN

Please input the start date (YYYY-MM-DD): 2020-01-01

Please input the end date (YYYY-MM-DD): 2020-12-31
[*********************100%***********************]  1 of 1 completed
The downloaded data is:
                   Open         High  ...    Adj Close   Volume
Date                                   ...
2020-01-02  1875.000000  1898.010010  ...  1898.010010  4029000
2020-01-03  1864.500000  1886.199951  ...  1874.969971  3764400
2020-01-06  1860.000000  1903.689941  ...  1902.880005  4061800
2020-01-07  1904.500000  1913.890015  ...  1906.859985  4044900
2020-01-08  1898.040039  1911.000000  ...  1891.969971  3508000
...                 ...          ...  ...          ...      ...
2020-11-30  3208.479980  3228.389893  ...  3168.040039  4063900
2020-12-01  3188.500000  3248.949951  ...  3220.080078  4544400
2020-12-02  3221.649902  3232.000000  ...  3203.530029  3129300
2020-12-03  3205.459961  3228.639893  ...  3186.729980  2892000
```

IPython console | History

LSP Python: ready | conda: base (Python 3.8.3) | Line 22, Col 28 | UTF-8 | CRLF | RW | Mem 90%

```
[235 rows x 6 columns]
Descriptive statistics for data is:                Open         High  ...    Adj
Close        Volume
count   235.000000   235.000000  ...   235.000000  2.350000e+02
mean   2641.316769  2677.899612  ...  2641.186210  5.036537e+06
std     551.911806   555.932445  ...   546.136295  1.987344e+06
min    1641.510010  1759.449951  ...  1676.609985  2.088000e+06
25%    2047.885010  2084.775024  ...  2064.755005  3.572050e+06
50%    2678.080078  2697.429932  ...  2675.010010  4.646300e+06
75%    3162.494995  3195.385010  ...  3148.444946  5.900150e+06
max    3547.000000  3552.250000  ...  3531.449951  1.556730e+07

[8 rows x 6 columns]
```

The CSV contains the data as well as the Descriptive statistics, so that the user can access it in a file.

6. Terms and Conditions

The application also provides an option to read the terms and conditions.
The terms and conditions are read from the text file in the folder by the application.

```
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): 6

*************************************************************************
Terms and Conditions for Vaishnavi's Stock Application.
This is a Text User Interface Application.
It provides the user with multiple options to work on, use and analyse the stock
data in an efficient manner.
The user prompts are written consicely and to the point.
*************************************************************************



Main Menu
1. Search Stocks
2. Query Time Range
3. Data Visualization
4. Stock Market Prediction
5. Display Descriptive Analytics on console and Download Data to CSV file
6. Terms and Conditions
0. Quit

Please choose an option (0 to exit): |
```

IPython console History

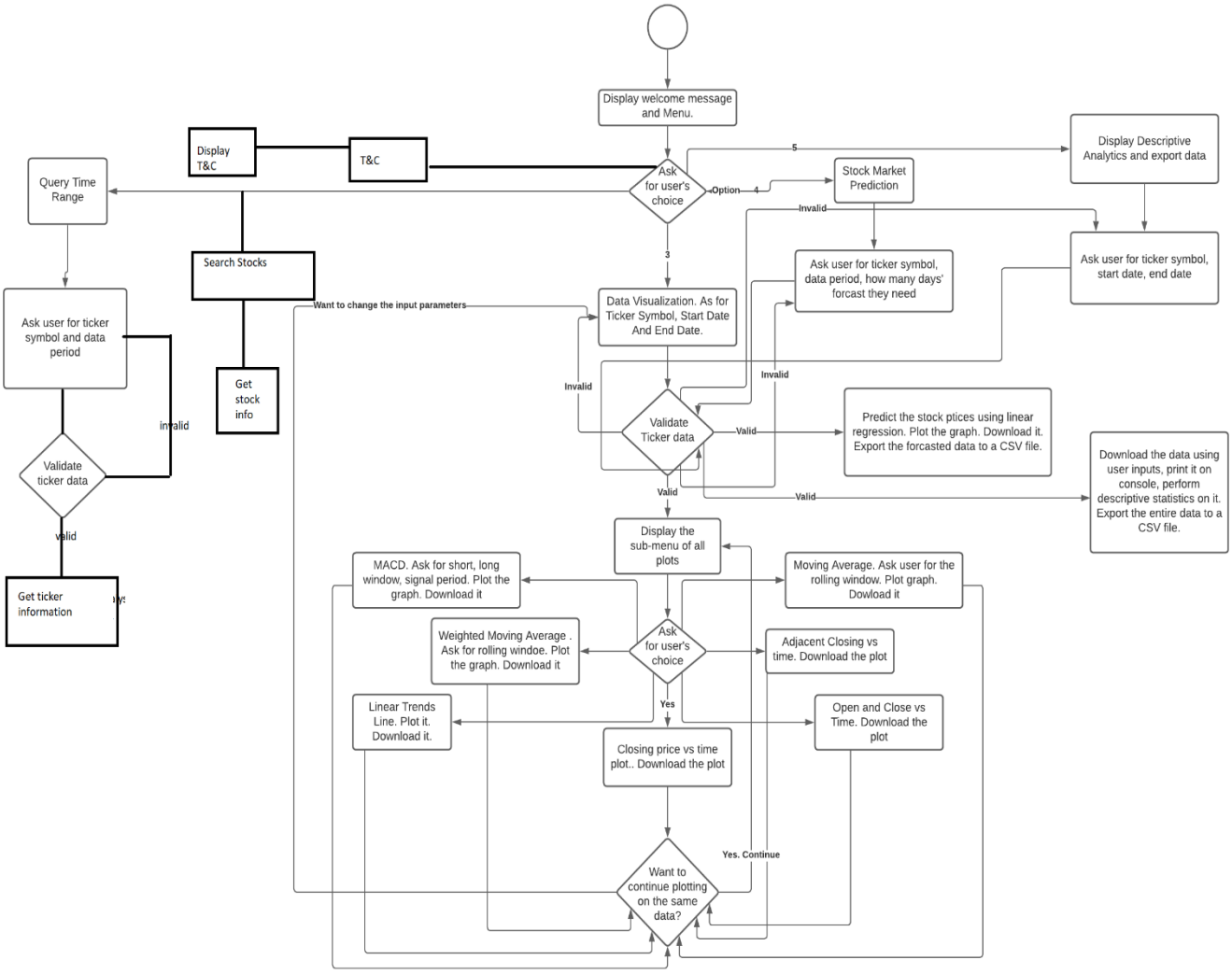
LSP Python: ready conda: base (Python 3.8.3) Line 139, Col 34 UTF-8 CRLF RW Mem 84%

The Quit button is 0. By pressing 0, the user can exit out of the application.

The structure of the code:

1. The code is divided into modules: The Data Visualization Logic is written completely in a different module, from where the Data Visualization Menu accesses the functions and calls the particular function depending upon what the user has inputted on the UI.
2. The Data Visualization Logic also contains the code to download the graph plot. This functionality is also interactive, where-in the application is asking the user if he/she wants to download the graph.
3. The Stock Predictor Logic is also an individual module, which is called in the Stock Quotes code, which is the main code where the main function initiates the application. If the user chooses to predict stock data, then it is called from the Stock Predictor Logic and gives the values on the UI, as well as exports them to a CSV and the graph is plot of the predicted values.
4. The Data Visualization Logic has 7 different definitions and they need the Ticker symbol, Start Date and End Date as parameters, so based on that the code plots the graphs. These parameters are being called by a single function get_ticker_and_date() which is being reused each time Ticker Symbol, Start Date and End Date is needed.
5. Validations on user input are made where the ticker symbol should be valid, date range must be valid, or else the user would be prompted.

UML DIAGRAM



The above UML Diagram gives a overview of how the application processes user input.

References:

1. Medium. 2020. *Building A Financial Trading Toolbox In Python: Simple Moving Average*. [online] Available at: <https://towardsdatascience.com/trading-toolbox-01-sma-7b8e16bd9388> [Accessed 6 December 2020].
2. Learndatasci.com. 2020. *Python For Finance, Part 3: Moving Average Trading Strategy*. [online] Available at: <https://www.learndatasci.com/tutorials/python-finance-part-3-moving-average-trading-strategy/> [Accessed 6 December 2020].
3. Medium. 2020. *In 12 Minutes: Stocks Analysis With Pandas And Scikit-Learn*. [online] Available at: <https://towardsdatascience.com/in-12-minutes-stocks-analysis-with-pandas-and-scikit-learn-a8d8a7b50ee7> [Accessed 6 December 2020].
4. Medium. 2020. *Free Stock Data For Python Using Yahoo Finance API*. [online] Available at: <https://towardsdatascience.com/free-stock-data-for-python-using-yahoo-finance-api-9dafd96cad2e> [Accessed 6 December 2020]