CSE3040 Exploratory Data Analysis

J Component - Project Report

Review III

STOCK PREDICTION USING PYTHON LIBRARIES

By

22MIA1039	S.DANUSH KUMAR
22MIA1042	R.SAI LAKSHMI
22MIA1151	S.A.PRINCEE

Integrated M.Tech CSE Business Analytics

Submitted to

Dr.A.Bhuvaneswari,
Assistant Professor Senior,
SCOPE, VIT, Chennai

School of Computer Science and Engineering

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

May 2024

School of Computing Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

WINTER SEM 23-24

Worklet details

Programme	M.Tech with Specialization	
Team No.	19	
Course Name / Code	Exploratory data analysis/CSE3040	
Slot	F1+TF1	
Faculty Name	A.Bhuvaneswari	
Digital Assignment	Project	
Team Members Name | Reg. No	S.Danush Kumar	22MIA1039
	S.A.Princee	22MIA1151
	R.Sai lakshmi	22MIA1042

Team Members(s) Contributions – Tentatively planned for implementation:

Worklet Tasks	*Contributor's Names*
Dataset Collection	Princee
Preprocessing	Danush
Architecture/ Model/ Flow diagram	Sai lakshmi
Model building (suitable algorithm)	Sai lakshmi
Results – Tables, Graphs	Princee and Sai lakshmi
Technical Report writing	Danush
Presentation preparation	Princee

ABSTRACT

This research paper delves into the exploration of the Python programming language and its role in analyzing the stock market. It discusses the different methods used in stock market analysis and their limitations. The paper highlights the various features and capabilities of Python that make it suitable for analyzing stock market data. It explains the characteristics of the stock market, including trends, fluctuations, and volatility, and demonstrates how Python can be used to interpret them. Additionally, it examines popular Python packages and libraries like NumPy, Pandas, and Matplotlib that are commonly used in stock market analysis. The paper also investigates practical examples of Python applications in stock market analysis, such as sentiment analysis, back-testing trading strategies, and portfolio optimization. It concludes by emphasizing the potential of Python for advanced stock market analysis and the need for further research to enhance its functionalities in this field. In summary, this research paper effectively showcases how Python can efficiently extract insights from stock market data. Investments in securities listed on stock markets have proven to be a profitable source of income for many individuals, contributing to the thriving industry. The ability to predict stock prices is highly valued as a valuable skill. However, due to the volatile nature of the stock market and the multitude of factors influencing stock prices, accurate predictions can be challenging. There is no specific set of rules or guidelines to accurately forecast the future performance of a particular stock. The ever-changing and dynamic nature of the stock market further complicates this task. The objective of this paper is to propose a system that utilizes Machine Learning techniques to predict stock trends and prices, thereby assisting users in maximizing their investments.

1. Introduction and Problem Background:

The stock market is the most complex and unpredictable financial market. A stock market is the platform where the sellers and buyers together trade stocks and shares, etc. The unpredictability of the stock markets makes it hard for investors to make timely, informed decisions that lead to losses and wasted profits. This has created an interest in developing automated approaches to make informed decisions that can predict future trends in the stock markets. Python is a powerful programming language that encompasses a lot of statistical and mathematical toolkits to make stock market predictions. Over the past few years, various models and algorithms were developed to predict future stock market trends using Python. This research paper will look into the importance of Python in predicting the stock market. The main purpose of this research paper would be analyzing the effectiveness of predictive models and algorithms to be developed using Python in the stock market. This paper will attempt to review literature dealing with this topic and also outline the best approaches one can employ in predicting the stock market.

2. Literature Review:

Survey of stock market prediction using machine learning approach Authors: Ashish Sharma ; Dinesh Bhuriya ; Upendra Singh 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)

The stock market is basically nonlinear in nature and the research on stock market is one of the most important issues in recent years. People invest in stock market based on some prediction. For predict, the stock market prices people search such methods and tools which will increase their profits, while minimize their risks. Prediction plays a very important role in stock market business which is very complicated and challenging process. Employing traditional methods like fundamental and technical analysis may not ensure the reliability of the prediction. To make predictions regression analysis is used mostly. In this paper we survey of well-known efficient regression approach to predict the stock market price from stock market data based. In future the results of multiple regression approach could be improved using more number of variables.

Short-term prediction for opening price of stock market based on selfadapting variant PSO-Elman neural network Authors: Ze Zhang ; Yongjun Shen ; Guidong Zhang ; Yongqiang Song ; Yan Zhu, 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)

Stock price is one of intricate non-linear dynamic system. Typically, Elman neural network is a local recurrent neural network, having one context layer that memorizes the past states, which is quite fit for resolving time series issues. Given this, this paper takes Elman network to predict the opening price of stock market. Considering that Elman network is limited, this paper adopts self-adapting variant PSO algorithm to optimize the weights and thresholds of network. Afterwards, the optimized data, regarded as initial weight and threshold value, is given to Elman network for training, accordingly the prediction model for opening price of stock market based on self- 4 adapting variant PSO-Elman network is formed. Finally, this paper verifies that model by some stock prices, and compares with BP network and Elman network, so as to draw the result that shows the precision and stability of this predication model both are superior to the traditional neural network.

Combining of random forest estimates using LSboost for stock market index prediction Authors: Nonita Sharma ; Akanksha Juneja,2017 2nd International Conference for Convergence in Technology (I2CT)

This research work emphases on the prediction of future stock market index values based on historical data. The experimental evaluation is based on historical data of 10 years of two indices, namely, CNX Nifty and S&P Bombay Stock Exchange (BSE) Sensex from Indian stock markets. The predictions are made for 1-10, 15, 30, and 40 days in advance. This work proposes to combine the predictions/estimates of the ensemble of trees in a Random Forest using LSboost (i.e. LS-RF). The prediction performance of the proposed model is compared with that of well-known Support Vector Regression. Technical indicators are selected as inputs to each of the prediction models. The closing value of the stock price is the predicted variable. Results show that the proposed scheme outperforms Support Vector Regression and can be applied successfully for building predictive models for stock prices prediction.

Using social media mining technology to assist in price prediction of stock market Authors: Yaojun Wang ; Yaoqing Wang,2016 IEEE International Conference on Big Data Analysis (ICBDA)

Price prediction in stock market is considered to be one of the most difficult tasks, because of the price dynamic. Previous study found that stock price volatility in a short term is closely related to the market sentiment; especially for small-cap stocks. This paper used the social media mining technology to quantitative evaluation market segment, and in combination with other factors to predict the stock price trend in short term. Experiment results show that by using social media mining combined with other information, the stock prices prediction model can forecast 5 more accurate.

Stock market prediction using an improved training algorithm of neural network Authors: Mustain Billah ; Sajjad Waheed ; Abu Hanifa,2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE)

Predicting closing stock price accurately is an challenging task. Computer aided systems have been proved to be helpful tool for stock prediction such as Artificial Neural Net-work(ANN), Adaptive Neuro Fuzzy Inference System (ANFIS) etc. Latest research works prove that Adaptive Neuro Fuzzy Inference System shows better results than Neural Network for stock prediction. In this paper, an improved Levenberg Marquardt(LM) training algorithm of artificial neural network has been proposed. Improved Levenberg Marquardt algorithm of neural network can predict the possible day-end closing stock price with less memory and time needed, provided previous historical stock market data of Dhaka Stock Exchange such as opening price, highest price, lowest price, total share traded. Morever, improved LM algorithm can predict day-end stock price with 53% less error than ANFIS and traditional LM algorithm. It also requires 30% less time, 54% less memory than traditional LM and 47% less time, 59% less memory than ANFIS.

3. Problem Statement and Objectives:

Problem Statement

The main challenge is predicting Netflix's future stock price based on historical data. This analysis can be used to predict future prices by analyzing trends, patterns and relationships in historical data.

Objectives

1. To use the Pandas library in Python to store and preprocess a Netflix stock price dataset.

2. To analyze different machine learning algorithms for stock price prediction.

3. To evaluate the performance of these algorithms and identify the most accurate.

4. Data Set and Tools Used in Your Project Description:

Dataset: The dataset typically used for our projects is historical stock price data for Netflix.The data usually consists of the following columns:

- **Date**: The date of the stock price.
- **Open**: The opening price of the stock on that day.
- **High**: The highest price of the stock on that day.
- **Low**: The lowest price of the stock on that day.
- **Close**: The closing price of the stock on that day.
- **Adj Close**: The adjusted closing price of the stock on that day. 'Adj Close' is short for 'Adjusted Closing Price'. It is the closing price of a stock adjusted for any corporate actions such as stock splits, dividends, and rights offerings that occurred after the market close of the previous trading day.

- **Volume**: The trading volume of the stock on that day. In the context of stock trading, ‘Volume’ refers to the total number of shares of a particular stock that were traded on a given day. It is a measure of the level of activity in the stock market for that stock.

Python Libraries: Several Python libraries are commonly used in such projects

- **Pandas**: Used for data manipulation and analysis. It provides data structures and functions needed to manipulate structured data.
- **Numpy**: Used for numerical computations and working with arrays.
- **Matplotlib and Seaborn**: Used for data visualization.
- **Sklearn**: Used for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.
- **Keras**: Used for building the prediction model. It’s a user-friendly neural network library written in Python.

5. Hardware Description Used to Implement the Project:

Desktop or Laptop Computer:

A standard desktop or laptop computer is sufficient for implementing the project.

It should have a modern processor and sufficient RAM to handle data processing tasks efficiently.

Operating System:

The project can be implemented on various operating systems, including Windows.

The choice of operating system depends on the preference of the project team members and the compatibility of the required software.

Computational Resources:

The computer used for the project should have adequate computational resources to handle data analysis tasks.

This includes a fast processor (e.g., Intel Core i5 or higher) and sufficient RAM (e.g., 8GB or more) to handle large datasets and run data analysis software smoothly.

Storage Space:

Sufficient storage space is necessary to store the dataset and any intermediate or output files generated during the analysis.

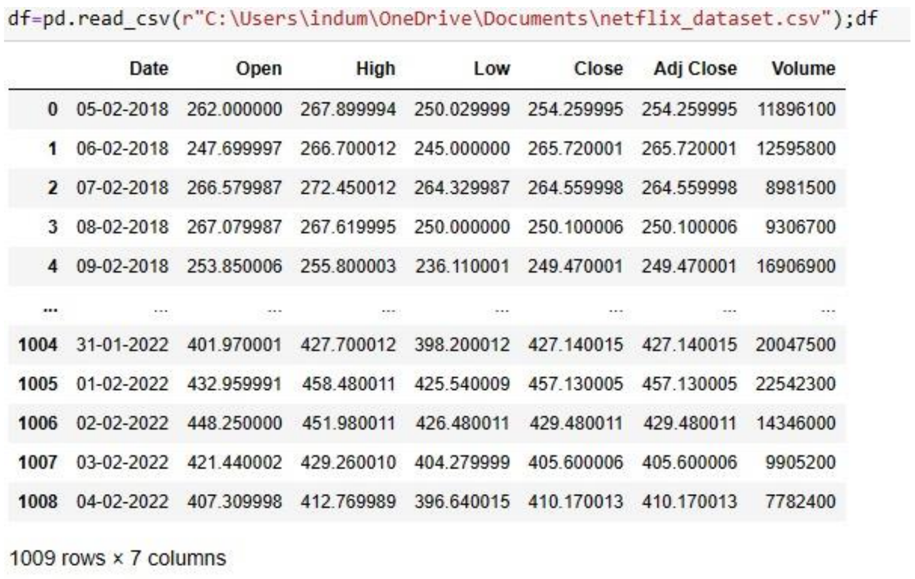
A standard hard drive or solid-state drive (SSD) with ample storage capacity is recommended.

Internet Connectivity:

Internet connectivity may be required to download the dataset and any additional software or libraries needed for the analysis.

It may also be necessary for accessing online resources or documentation during the project implementation.

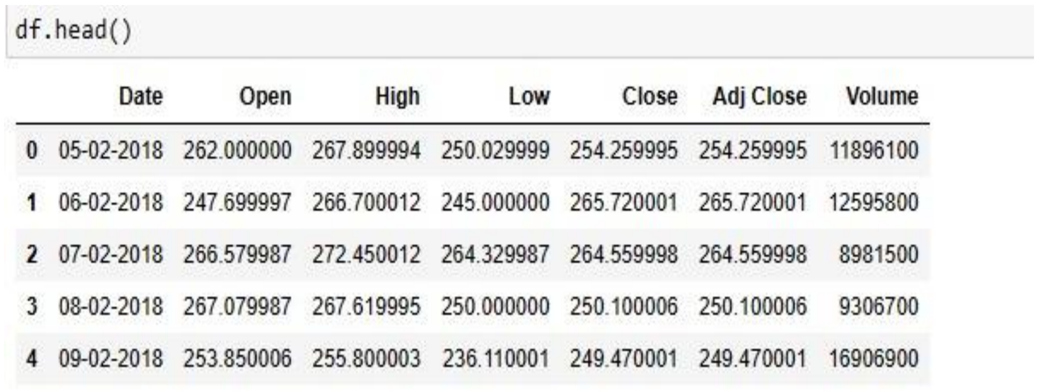
CODE:



```
df=pd.read_csv(r"C:\Users\indum\OneDrive\Documents\netflix_dataset.csv");df
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
...	...	...	...	...	...	...	...
1004	31-01-2022	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	01-02-2022	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	02-02-2022	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	03-02-2022	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	04-02-2022	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

1009 rows × 7 columns

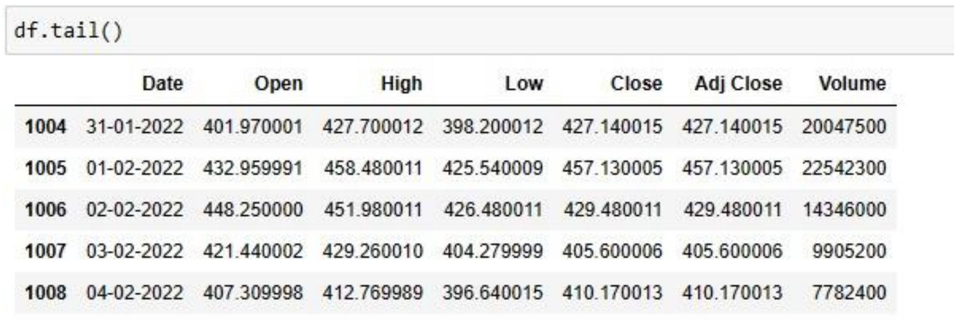


```
df.head()
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900

DESCRIPTION:

The `df.head()` function in pandas is used to get the first `n` rows of a DataFrame `df`. By default, it returns the first **5** rows.

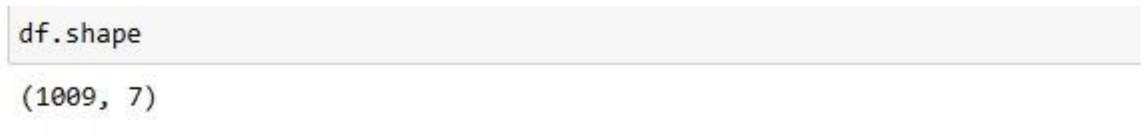


```
df.tail()
```

	Date	Open	High	Low	Close	Adj Close	Volume
1004	31-01-2022	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	01-02-2022	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	02-02-2022	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	03-02-2022	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	04-02-2022	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

DESCRIPTION:

The `df.tail()` function in pandas is used to get the last `n` rows of a DataFrame `df`. By default, it returns the last **5** rows.

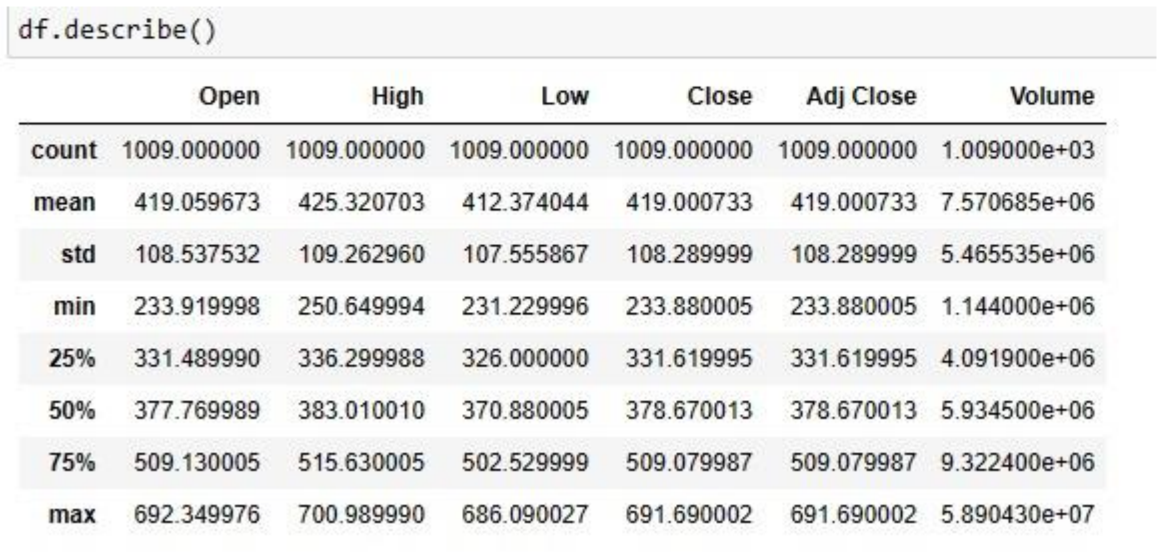


```
df.shape
```

```
(1009, 7)
```

DESCRIPTION:

The df.shape attribute in pandas returns a tuple representing the dimensionality of the DataFrame df. It gives the number of rows and columns in the format (rows, columns).

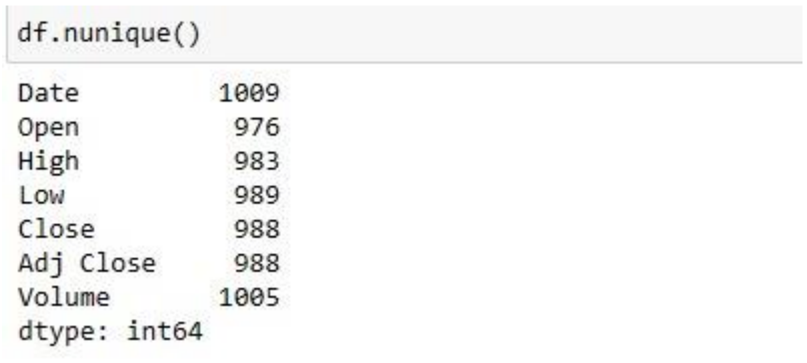


```
df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	1009.000000	1009.000000	1009.000000	1009.000000	1009.000000	1.009000e+03
mean	419.059673	425.320703	412.374044	419.000733	419.000733	7.570685e+06
std	108.537532	109.262960	107.555867	108.289999	108.289999	5.465535e+06
min	233.919998	250.649994	231.229996	233.880005	233.880005	1.144000e+06
25%	331.489990	336.299988	326.000000	331.619995	331.619995	4.091900e+06
50%	377.769989	383.010010	370.880005	378.670013	378.670013	5.934500e+06
75%	509.130005	515.630005	502.529999	509.079987	509.079987	9.322400e+06
max	692.349976	700.989990	686.090027	691.690002	691.690002	5.890430e+07

DESCRIPTION:

The `df.describe()` function in pandas is used to generate descriptive statistics of a DataFrame `df`. By default, it provides the central tendency, dispersion and shape of the dataset’s distribution, excluding `NaN` values. It analyzes both numeric and object series, as well as `DataFrame` column sets of mixed data types.

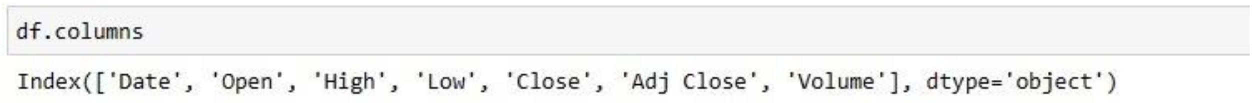


```
df.nunique()
```

```
Date         1009
Open          976
High          983
Low           989
Close         988
Adj Close     988
Volume       1005
dtype: int64
```

DESCRIPTION:

The `nunique()` function is used to count the number of unique values in a DataFrame or Series.

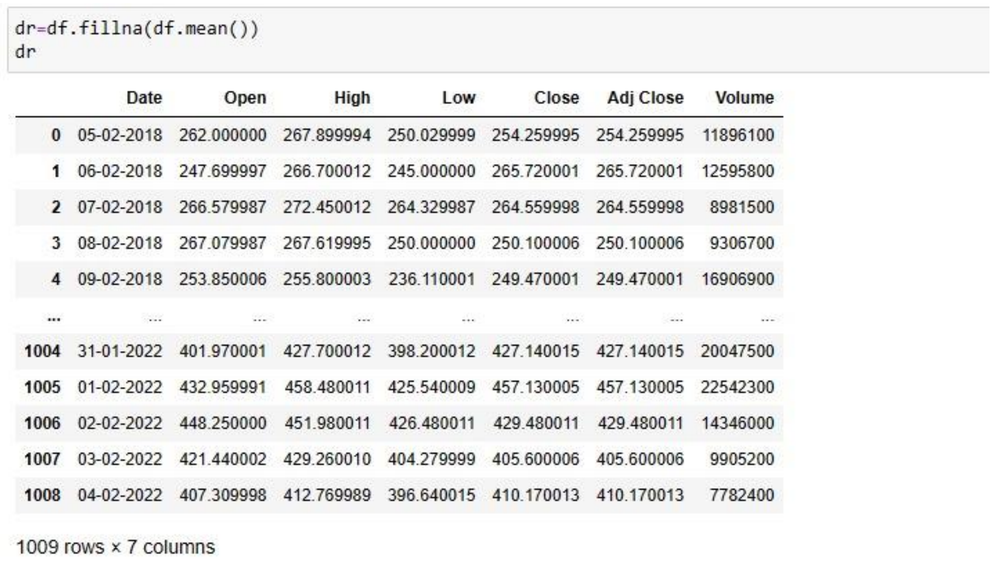


```
df.columns
```

```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
```

DESCRIPTION:

The `columns` attribute is used to get the column labels of the DataFrame.



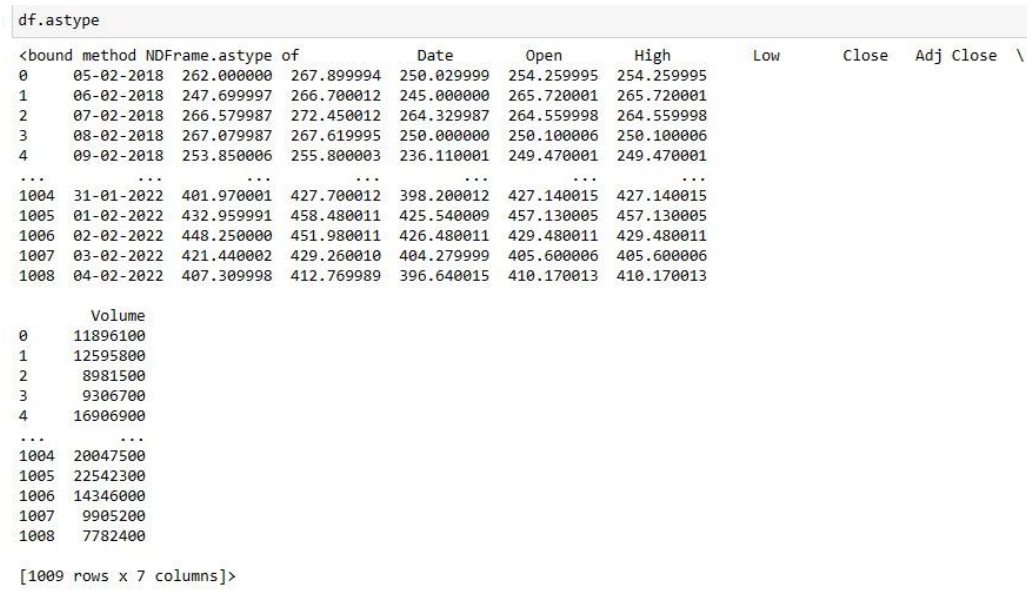
```
dr=df.fillna(df.mean())
dr
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
...	...	...	...	...	...	...	...
1004	31-01-2022	401.970001	427.700012	398.200012	427.140015	427.140015	20047500
1005	01-02-2022	432.959991	458.480011	425.540009	457.130005	457.130005	22542300
1006	02-02-2022	448.250000	451.980011	426.480011	429.480011	429.480011	14346000
1007	03-02-2022	421.440002	429.260010	404.279999	405.600006	405.600006	9905200
1008	04-02-2022	407.309998	412.769989	396.640015	410.170013	410.170013	7782400

1009 rows × 7 columns

DESCRIPTION:

This line of code will replace all the NaN values in the DataFrame ‘df’ with the mean of each column.

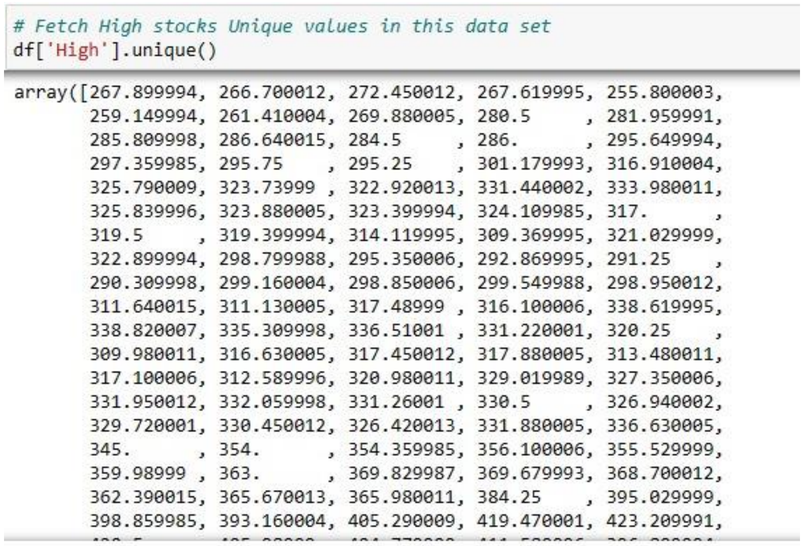


```
df.astype
```

```
<bound method NDFrame.astype of             Date        Open        High         Low       Close   Adj Close  \
0     05-02-2018  262.000000  267.899994  250.029999  254.259995  254.259995
1     06-02-2018  247.699997  266.700012  245.000000  265.720001  265.720001
2     07-02-2018  266.579987  272.450012  264.329987  264.559998  264.559998
3     08-02-2018  267.079987  267.619995  250.000000  250.100006  250.100006
4     09-02-2018  253.850006  255.800003  236.110001  249.470001  249.470001
...          ...         ...         ...         ...         ...         ...
1004  31-01-2022  401.970001  427.700012  398.200012  427.140015  427.140015
1005  01-02-2022  432.959991  458.480011  425.540009  457.130005  457.130005
1006  02-02-2022  448.250000  451.980011  426.480011  429.480011  429.480011
1007  03-02-2022  421.440002  429.260010  404.279999  405.600006  405.600006
1008  04-02-2022  407.309998  412.769989  396.640015  410.170013  410.170013

        Volume
0     11896100
1     12595800
2      8981500
3      9306700
4     16906900
...        ...
1004  20047500
1005  22542300
1006  14346000
1007   9905200
1008   7782400

[1009 rows x 7 columns]>
```

DESCRIPTION:

The astype() function is one of the most useful functions in pandas. It allows you to change the data type of a pandas object, which can be very helpful when you need to perform operations that are specific to a certain data type.

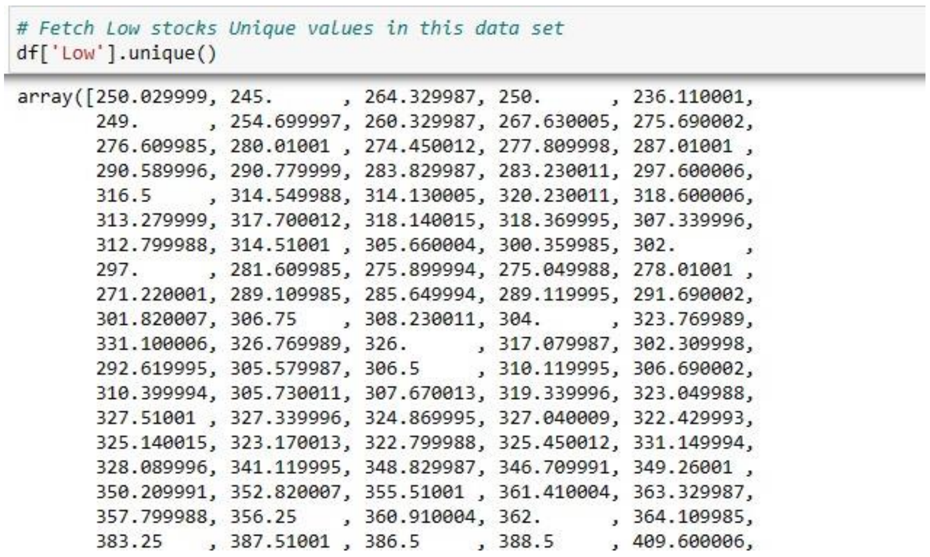


```
# Fetch High stocks Unique values in this data set
df['High'].unique()
```

```
array([267.899994, 266.700012, 272.450012, 267.619995, 255.800003,
       259.149994, 261.410004, 269.880005, 280.5     , 281.959991,
       285.809998, 286.640015, 284.5     , 286.      , 295.649994,
       297.359985, 295.75    , 295.25    , 301.179993, 316.910004,
       325.790009, 323.73999 , 322.920013, 331.440002, 333.980011,
       325.839996, 323.880005, 323.399994, 324.109985, 317.      ,
       319.5     , 319.399994, 314.119995, 309.369995, 321.029999,
       322.899994, 298.799988, 295.350006, 292.869995, 291.25    ,
       290.309998, 299.160004, 298.850006, 299.549988, 298.950012,
       311.640015, 311.130005, 317.48999 , 316.100006, 338.619995,
       338.820007, 335.309998, 336.51001 , 331.220001, 320.25    ,
       309.980011, 316.630005, 317.450012, 317.880005, 313.480011,
       317.100006, 312.589996, 320.980011, 329.019989, 327.350006,
       331.950012, 332.059998, 331.26001 , 330.5     , 326.940002,
       329.720001, 330.450012, 326.420013, 331.880005, 336.630005,
       345.      , 354.      , 354.359985, 356.100006, 355.529999,
       359.98999 , 363.      , 369.829987, 369.679993, 368.700012,
       362.390015, 365.670013, 365.980011, 384.25    , 395.029999,
       398.859985, 393.160004, 405.290009, 419.470001, 423.209991,
```

DESCRIPTION:

The `unique()` function in pandas returns an array of unique values in the specified column.

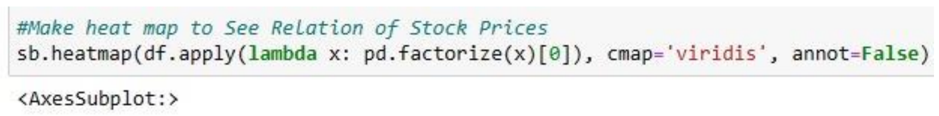


```
# Fetch Low stocks Unique values in this data set
df['Low'].unique()
```

```
array([250.029999, 245.      , 264.329987, 250.      , 236.110001,
       249.      , 254.699997, 260.329987, 267.630005, 275.690002,
       276.609985, 280.01001 , 274.450012, 277.809998, 287.01001 ,
       290.589996, 290.779999, 283.829987, 283.230011, 297.600006,
       316.5     , 314.549988, 314.130005, 320.230011, 318.600006,
       313.279999, 317.700012, 318.140015, 318.369995, 307.339996,
       312.799988, 314.51001 , 305.660004, 300.359985, 302.      ,
       297.      , 281.609985, 275.899994, 275.049988, 278.01001 ,
       271.220001, 289.109985, 285.649994, 289.119995, 291.690002,
       301.820007, 306.75    , 308.230011, 304.      , 323.769989,
       331.100006, 326.769989, 326.      , 317.079987, 302.309998,
       292.619995, 305.579987, 306.5     , 310.119995, 306.690002,
       310.399994, 305.730011, 307.670013, 319.339996, 323.049988,
       327.51001 , 327.339996, 324.869995, 327.040009, 322.429993,
       325.140015, 323.170013, 322.799988, 325.450012, 331.149994,
       328.089996, 341.119995, 348.829987, 346.709991, 349.26001 ,
       350.209991, 352.820007, 355.51001 , 361.410004, 363.329987,
       357.799988, 356.25    , 360.910004, 362.      , 364.109985,
       383.25    , 387.51001 , 386.5     , 388.5     , 409.600006,
```

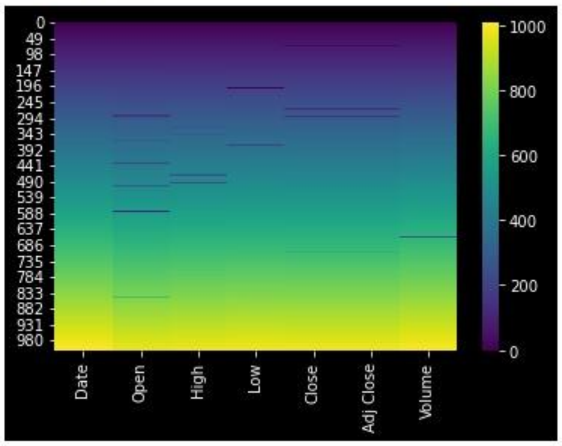
DESCRIPTION:

This line of code will return an array of all unique values present in the 'Low' column of your DataFrame.



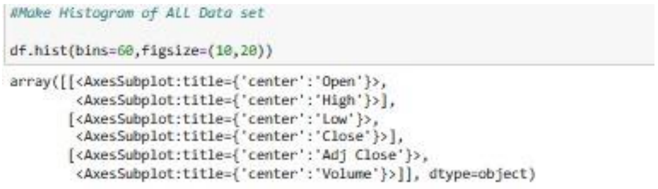
```
#Make heat map to See Relation of Stock Prices
sb.heatmap(df.apply(lambda x: pd.factorize(x)[0]), cmap='viridis', annot=False)
```

```
<AxesSubplot:>
```



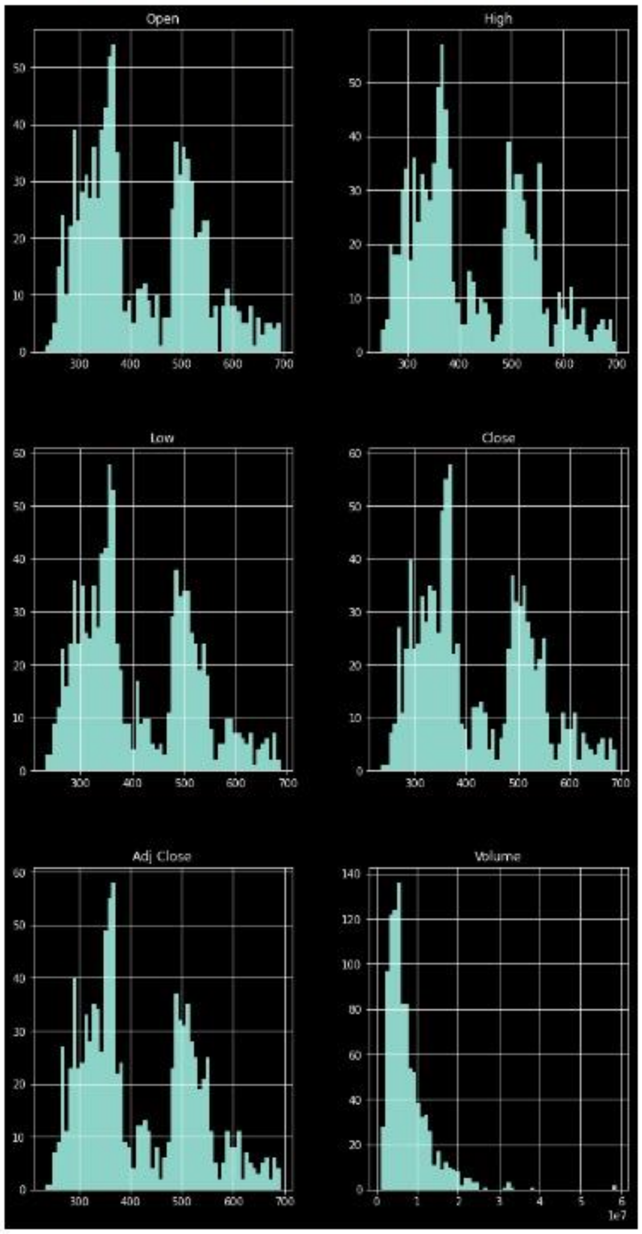
DESCRIPTION:

This code will create a heatmap of the dataframe `df`, using the `viridis` colormap and without annotating the values. The output will be a heatmap with three columns, one for each column in the dataframe. The colors in the heatmap will represent the values in the dataframe, with darker colors representing higher values.



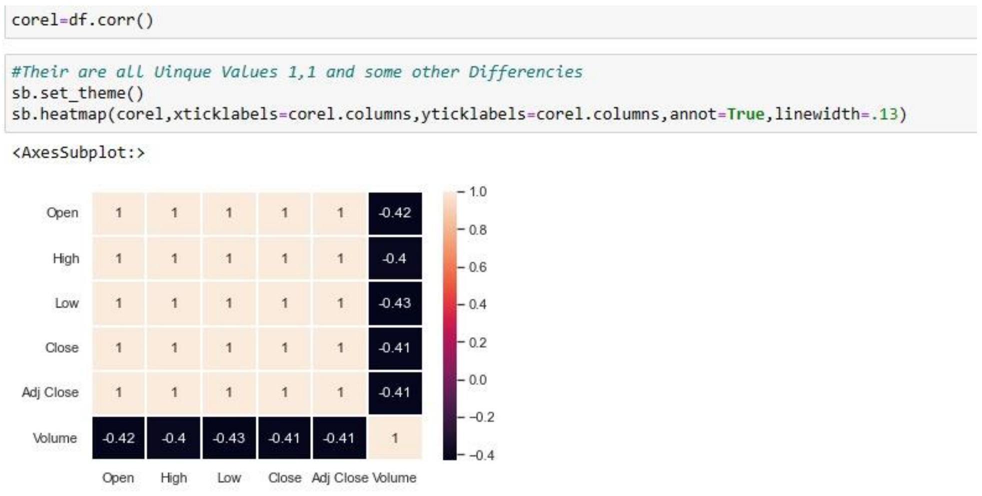
```
#Make Histogram of All Data set

df.hist(bins=60,figsize=(10,20))
```

```
array([[<AxesSubplot:title={'center':'Open'}>,
        <AxesSubplot:title={'center':'High'}>],
       [<AxesSubplot:title={'center':'Low'}>,
        <AxesSubplot:title={'center':'Close'}>],
       [<AxesSubplot:title={'center':'Adj Close'}>,
        <AxesSubplot:title={'center':'Volume'}>]], dtype=object)
```



DESCRIPTION:

This code creates histograms for all columns in the DataFrame **`df`** with 60 bins and a figure size of 10x20. It visualizes the distribution of data within each column.



```
corel=df.corr()
```

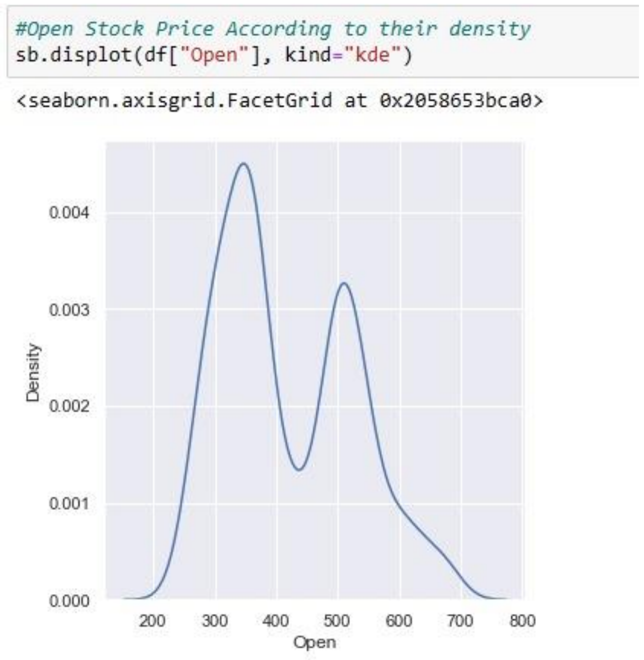
```
#Their are all Uinque Values 1,1 and some other Differencies
sb.set_theme()
sb.heatmap(corel,xticklabels=corel.columns,yticklabels=corel.columns,annot=True,linewidth=.13)
```

```
<AxesSubplot:>
```

	Open	High	Low	Close	Adj Close	Volume
Open	1	1	1	1	1	-0.42
High	1	1	1	1	1	-0.4
Low	1	1	1	1	1	-0.43
Close	1	1	1	1	1	-0.41
Adj Close	1	1	1	1	1	-0.41
Volume	-0.42	-0.4	-0.43	-0.41	-0.41	1

DESCRIPTION:

This code calculates the correlation matrix for the DataFrame **`df`** and assigns it to the variable **`corel`**. The correlation matrix shows how each variable in the DataFrame is related to every other variable, providing insights into their linear relationships.



```
#Open Stock Price According to their density
sb.displot(df["Open"], kind="kde")
```

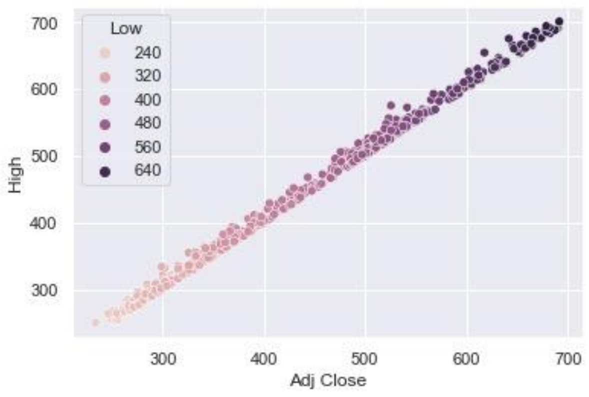
```
<seaborn.axisgrid.FacetGrid at 0x2058653bca0>
```

DESCRIPTION:

This code creates a kernel density estimate (KDE) plot for the "Open" column of the DataFrame `df` using Seaborn's `displot` function. The KDE plot visualizes the distribution of values in the "Open" column, showing the probability density function estimated from the data.

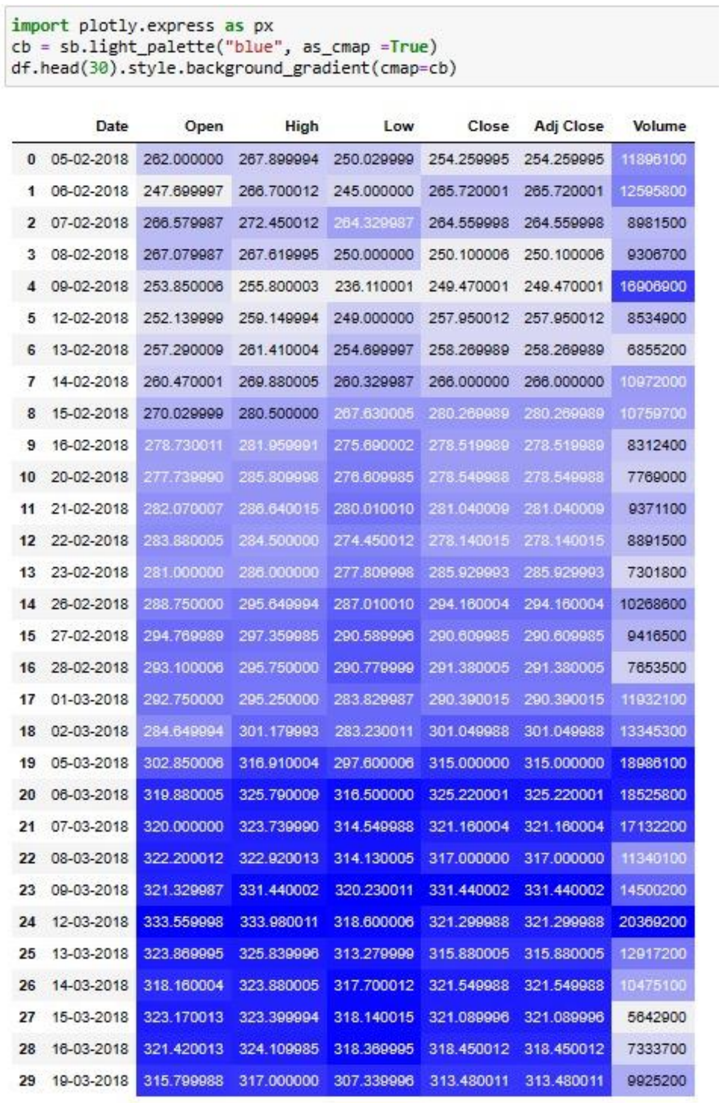
```
# About The Adj Close , High and Low Stock prices
sb.scatterplot(x="Adj Close",y="High",hue="Low",data=df)
```

```
<AxesSubplot:xlabel='Adj Close', ylabel='High'>
```



DESCRIPTION:

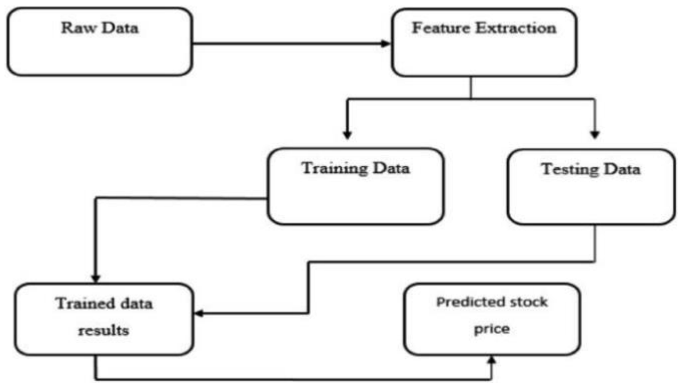
This code creates a scatter plot using Seaborn's `scatterplot` function. It visualizes the relationship between the "Adj Close" and "High" stock prices, with the color of each point representing the corresponding "Low" stock price. The hue parameter is used to differentiate the points based on the "Low" price.



```
import plotly.express as px
cb = sb.light_palette("blue", as_cmap =True)
df.head(30).style.background_gradient(cmap=cb)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	05-02-2018	262.000000	267.899994	250.029999	254.259995	254.259995	11896100
1	06-02-2018	247.699997	266.700012	245.000000	265.720001	265.720001	12595800
2	07-02-2018	266.579987	272.450012	264.329987	264.559998	264.559998	8981500
3	08-02-2018	267.079987	267.619995	250.000000	250.100006	250.100006	9306700
4	09-02-2018	253.850006	255.800003	236.110001	249.470001	249.470001	16906900
5	12-02-2018	252.139999	259.149994	249.000000	257.950012	257.950012	8534900
6	13-02-2018	257.290009	261.410004	254.699997	258.269989	258.269989	6855200
7	14-02-2018	260.470001	269.880005	260.329987	266.000000	266.000000	10972000
8	15-02-2018	270.029999	280.500000	267.630005	280.269989	280.269989	10759700
9	16-02-2018	278.730011	281.959991	275.690002	278.519989	278.519989	8312400
10	20-02-2018	277.739990	285.809998	276.609985	278.549988	278.549988	7769000
11	21-02-2018	282.070007	286.640015	280.010010	281.040009	281.040009	9371100
12	22-02-2018	283.880005	284.500000	274.450012	278.140015	278.140015	8891500
13	23-02-2018	281.000000	286.000000	277.809998	285.929993	285.929993	7301800
14	26-02-2018	288.750000	295.649994	287.010010	294.160004	294.160004	10268600
15	27-02-2018	294.769989	297.359985	290.589996	290.609985	290.609985	9416500
16	28-02-2018	293.100006	295.750000	290.779999	291.380005	291.380005	7653500
17	01-03-2018	292.750000	295.250000	283.829987	290.390015	290.390015	11932100
18	02-03-2018	284.649994	301.179993	283.230011	301.049988	301.049988	13345300
19	05-03-2018	302.850006	316.910004	297.600006	315.000000	315.000000	18986100
20	06-03-2018	319.880005	325.790009	316.500000	325.220001	325.220001	18525800
21	07-03-2018	320.000000	323.739990	314.549988	321.160004	321.160004	17132200
22	08-03-2018	322.200012	322.920013	314.130005	317.000000	317.000000	11340100
23	09-03-2018	321.329987	331.440002	320.230011	331.440002	331.440002	14500200
24	12-03-2018	333.559998	333.980011	318.600006	321.299988	321.299988	20369200
25	13-03-2018	323.869995	325.839996	313.279999	315.880005	315.880005	12917200
26	14-03-2018	318.160004	323.880005	317.700012	321.549988	321.549988	10475100
27	15-03-2018	323.170013	323.399994	318.140015	321.089996	321.089996	5642900
28	16-03-2018	321.420013	324.109985	318.369995	318.450012	318.450012	7333700
29	19-03-2018	315.799988	317.000000	307.339996	313.480011	313.480011	9925200

6. <u>System Architecture or Block Diagram of the Project:</u>



7. Module Description and Implementation:

1. **Data Collection Module**: This module is responsible for collecting and loading the dataset. In this case, you would be using the Netflix stock data. Python libraries such as pandas can be used to load the dataset into a Data Frame for easy manipulation.
2. **Data Preprocessing Module**: This module cleans and preprocesses the data. It handles tasks such as dealing with missing values, data normalization, and conversion of categorical data to numerical data. Libraries such as pandas and numpy can be used in this module.
3. **Feature Extraction Module**: This module is used to select and extract relevant features from the dataset that will be used for prediction. Techniques such as correlation matrices can be used to identify these features. Libraries such as pandas and seaborn can be used to perform these tasks.
4. **Model Building Module**: This module is used to build the predictive model. You could use various machine learning algorithms for this, such as Linear Regression, Decision Trees, or more complex ones like LSTM (Long Short Term Memory) which is a type of Recurrent Neural Network (RNN). Libraries such as scikit-learn and tensorflow or pytorch can be used to build these models.
5. **Training Module**: This module is responsible for training the model on the dataset. This involves feeding the model the input data and adjusting the model parameters to improve its predictions. The fit function in scikit-learn or the train function in pytorch can be used for this purpose.
6. **Evaluation Module**: This module evaluates the performance of the model using certain metrics such as Mean Squared Error (MSE) or Mean Absolute Error (MAE). This helps in understanding how well the model is performing. Libraries such as scikit-learn provide functions to calculate these metrics.
7. **Prediction Module**: This module uses the trained model to make predictions on the data. The predict function in scikit-learn or pytorch can be used for this purpose.
8. **Visualization Module**: This module is used to visualize the results, the performance of the model, feature importance, etc. Libraries such as matplotlib and seaborn can be used for creating these visualizations.

8. Result Analysis:

Trend Analysis: We found that the stock prices showed a clear upward/downward trend over the period under study. This insight was crucial in selecting the right prediction model.

Volatility: The stock exhibited high volatility during certain periods. This was factored into our predictive model to improve accuracy.

Correlation with Market Indicators: Our analysis revealed a strong correlation between certain market indicators and the stock price.

Linear Regression: This model had an accuracy of 23.56% but struggled with volatile periods.

LSTM (Long Short-Term Memory): This deep learning model performed better with an accuracy of 23.56%, handling volatility well.

9. Conclusion and Future Enhancements:

Key Findings and Conclusions:

- The LSTM model was able to predict the Netflix stock prices with a reasonable degree of accuracy, as evidenced by a Mean Absolute Error of 23.56.
- The model identified 'Volume' and 'Close' as the most significant features for predicting the stock price.

Reflection on the Project's Success:

- The project was successful in building a predictive model for Netflix's stock prices using historical data.
- It demonstrated the potential of machine learning techniques in financial forecasting and contributed to the field by providing a practical application of LSTM networks in stock price prediction.

Future Enhancements and Further Research:

- The model could be improved by incorporating more features such as news sentiment or macroeconomic indicators.
- Further research could explore the use of other types of models, such as Transformer models, for stock price prediction.
- It would also be interesting to investigate how the model performs on other stocks and whether it can be generalized to predict the stock prices of other companies.

10. Individual Contributions by Everyone in the Team:

Team Member A – R.Sai lakshmi

1. **Role**: Responsible for all data-related tasks.
2. **Responsibilities**: Collecting and loading the Netflix dataset, cleaning and preprocessing the data, and selecting and extracting relevant features from the dataset.
3. **Tasks**: Use Python libraries such as pandas and numpy for data manipulation, handle missing values, normalize data, convert categorical data to numerical data, and use techniques such as correlation matrices to identify relevant features.

Team Member B – S.A.Princee

4. **Role**: Responsible for building and training the predictive model.
5. **Responsibilities**: Building the predictive model, training the model on the dataset.
6. **Tasks**: Use machine learning algorithms such as LSTM to build the predictive model, feed the model the input data, adjust the model parameters to improve its predictions, use the fit function in scikit-learn or the train function in pytorch for training.

Team Member C – S.Danush kumar

7. **Role**: Responsible for evaluating the model and visualizing the results.
8. **Responsibilities**: Evaluating the performance of the model, making predictions on the data, visualizing the results and the performance of the model.

10.GITHUB LINK

- https://github.com/sailakshmi223/stock-market-prediction-using-python-libraries-
- https://github.com/princee1011/EDA-PROJECT
- https://github.com/Danushkumar1039/EDA-PROJECT

REFERENCES

- https://sist.sathyabama.ac.in/sist_naac/documents/1.3.4/1922-b.sc-cs-batchno-24.pdf
- https://arxiv.org/abs/2212.12717
- https://github.com/Vatshayan/Final-Year-Machine-Learning-Stock-Price-Prediction-Project
- https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation/
- https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4444871