# GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY, TIRUTTANI - 631209

**Approved by AICTE, New Delhi Affiliated to Anna University, Chennai**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**STOCK PRICE PREDICTION**

PROJECT REPORT

SUBMITTED BY

**GOKULAVASAN M C**

**3RD YEAR 5TH SEM**

**110321104010**

**gokulavasan137@gmail.com**

# ANNA UNIVERSITY:CHENNAI 600 025

## BONAFIDE CERTIFICATE

Certified that this project report **"STOCK PRICE PREDICTION"** is the bonafide work of **"GOKULAVASAN M C [110321104010]"** who carried out the project work under my our supervision.

| | |
|---|---|
| **SIGNATURE** | **SIGNATURE** |
| **Dr.N. Kamal M.E.,Ph.D.,** | **Mr.T.A. Vinayagam M.Tech.,** |
| **HOD** | **Assistant professor** |
| Department of Computer Science And Engineering | Department of Computer Science And Engineering |
| GRT Institute of Engineering and Technology | GRT Institute of Engineering and Technology |
| Tiruttani | Tiruttani |

Certified that the candidates were examined in Viva-voce in the Examination Held on______________________

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# ACKNOWLEDGEMENT

We thank our Management for providing us all support to complete this project successfully. Our sincere thanks to honorable **Chairman**, **Shri. G. RAJENDRAN and Managing Director, Shri.G.R. RADHAKRISHNAN** for creating a wonderful atmosphere inside the campus.

We are very grateful to **Dr.S. ARUMUGAM, M.E., Ph.D., Principal**, for providing us with consistent guidance and motivation to execute a real time project to learn and experience the project work in an environments to complete our project successfully.

Our sincere thanks to **Dr.N. KAMAL, M.E., Ph.D., Professor and Head, Department of Computer Science and Engineering** for giving me this wonderful opportunity to do the projectand providing the require facilities to fulfill our work.

We are highly indebted and thankful to our project Evaluators **Mrs V. Priya M.E**., and **Mrs. Edith Esther M.E., Assistant Professor, Department of Computer Science andEngineering** for her immense support in doing the project.

We are very grateful to our internal guide **Mr. T.A. VINAYAGAM, M.Tech., Assistant Professor, Department of Computer Science and Engineering** for guiding us with his valuable suggestions to complete our project.

We also dedicate equal and grateful acknowledgement to all the **respectable members of the faculty and lab in-charges** of the Department of Computer Science and Engineering, friends andour families for their motivation, encouragement and continuous support.

Our sincere thanks **to IBM and Skill Up Team members** for giving me this wonderful opportunity to do the project and providing the require guidance and valuable online sessions.

# ABSTRACT

Stock price prediction is a critical area of research and application in financial markets, with substantial implications for investors, traders, and financial institutions. This abstract summarizes the key aspects of stock price prediction, highlighting the methods, challenges, and potential benefits associated with this field.

Stock price prediction involves the use of various techniques and models to forecast the future prices of publicly traded stocks. Machine learning and statistical models have gained prominence in recent years due to their ability to analyse vast datasets and capture complex patterns. Commonly employed models include time series analysis, regression analysis, artificial neural networks, and deep learning algorithms.

The challenges in stock price prediction are multifaceted. Factors such as market volatility, external events, and investor sentiment can significantly impact stock prices, making accurate predictions inherently difficult Additionally, overfitting and model evaluation are persistent challenges that require careful consideration.

Despite these challenges, stock price prediction offers several potential benefits. Investors can use predictive models to make informed decisions, optimize portfolio allocations, and manage risk. Traders can implement algorithmic trading strategies based on price forecasts to achieve better returns.

stock price prediction is a dynamic and evolving field that combines finance, data science, and machine learning to forecast future stock prices. While challenges persist, the potential benefits of accurate predictions make this area of research and application highly valuable in the context of financial markets. Ongoing advancements in technology and data analysis techniques continue to shape the future of stock price.

| | TABLE OF CONTENTS | |
|---|---|---|
| S.NO No | TITLE | PAGE No |
| | **ABSTRACT** | 4 |
| **1.** | **CHAPTER 1** | 7 |
| | 1.1 INTRODUCTION | 7 |
| | 1.2 PROBLEM DEFINITION | 7 |
| | 1.3 OBJECTIVES | 8 |
| | 1.4 DESIGN THINKING | 9 |
| | 1.5 SYSTEM ARCHITECTURE | 9 |
| | 1.6 E – R DIAGRAM | 9 |
| | 1.7 USE CASE DIAGRAM | 10 |
| | 1.8 BLOCK DIAGRAM | 11 |
| | | 12 |
| **2.** | **CHAPTER 2** | 13 |
| | 2.1 INNOVATION | 13 |
| | 2.2 DEEP LEARNING | 13 |
| | 2.2.1 USING NUMPY | 13 |
| | 2.2.2 USING PANDAS | 14 |
| | 2.3 LONG SHORT-TERM MEMORY NEURAL NETWORKS | 15 |

# CHAPTER 1

# INTRODUCTION

Stock Price Prediction System is a data-driven project aimed at forecasting the future prices of publicly traded stocks. This project leverages advanced machine learning and data analysis techniques to provide insights into stock market behaviour and assist investors, traders, and financial decisions. The heart of the system lies in model development, where a diverse range of predictive algorithms, including time series analysis, regression, machine learning, and deep learning, are considered and fine-tuned to optimize their predictive capabilities.

This system incorporates several critical components to achieve its objectives. It begins with data collection and preprocessing, where historical stock price data, trading volumes, and external factors such as economic indicators and news sentiment are meticulously curated and cleaned. Moreover, back testing enables the assessment of trading strategies rooted in these predictions.

The user experience is central to the system's design, with a user-friendly interface crafted for both web and mobile platforms. This interface empowers users to input specific stock symbols, select desired time horizons, and access predictions, historical data, and interactive visualizations. Strong emphasis is placed on security, with robust authentication mechanisms safeguarding user data and system integrity. Furthermore, the option for real-time data integration allows the system to continually update and remain responsive to dynamic market conditions.

## PROBLEM DEFINITION:

Predicting stock prices is a challenging endeavour due to the inherent complexity and volatility of financial markets. Here are some of the key problems and challenges that researchers and practitioners commonly face when attempting to predict stock prices:

1.**Market Volatility:** Financial markets are characterized by frequent and sometimes unpredictable price fluctuations. Sudden changes in market sentiment, external events, or macroeconomic factors can lead to rapid and significant price movements, making it challenging to accurately predict future prices.

2.**Data Quality and Noise:** Financial data, including stock prices and trading volumes, can be noisy and subject to errors. Missing data, data outliers, and discrepancies in reporting can impact the quality and reliability of historical data used for prediction.

3.**Feature Selection:** Determining which features or variables to include in predictive models is a critical challenge. Financial data is multidimensional, and selecting the most relevant factors that influence stock prices requires expertise and careful consideration.

4.**Overfitting and Generalization**: Developing predictive models that generalize well to unseen data is a constant challenge. Complex models can overfit to historical data, capturing noise rather than meaningful patterns, which can lead to poor performance when applied to new data.

5.**Market Efficiency:** The Efficient Market Hypothesis (EMH) suggests that stock prices already incorporate all publicly available information. This hypothesis implies that it is difficult to gain a consistent edge in predicting stock prices, as they are believed to be close to their intrinsic values. Thus, accurately predicting price deviations from these values is challenging

## OBJECTIVES:

The objectives of stock price prediction can vary depending on the stakeholders involved and the specific context in which the predictions are used. Here are some common objectives associated with stock price prediction:

1**.Investment Selection:** To assist investors in identifying stocks with the potential for capital appreciation or income generation, allowing them to make informed decisions about buying or holding specific stocks.

2.**Risk Mitigation**: To help investors and portfolio managers assess and manage risks associated with stock investments by predicting potential price downturns or fluctuations.

3.**Portfolio Diversification:** To optimize portfolio allocation by selecting a mix of stocks that balance risk and return, aiming for a diversified portfolio that maximizes returns for a given level of risk.

4.**Trading Strategies:** To develop and implement trading strategies that leverage short-term price movements, including strategies based on technical analysis, algorithmic trading, or high-frequency trading.

5.**Market Timing:** To identify opportune moments to enter or exit the market or specific stocks, helping traders and investors make timely decisions to maximize gains or minimize losses.

6.**Performance Evaluation:** To assess the effectiveness of investment strategies, asset managers, or trading algorithms by comparing actual stock price movements to predicted price trends.

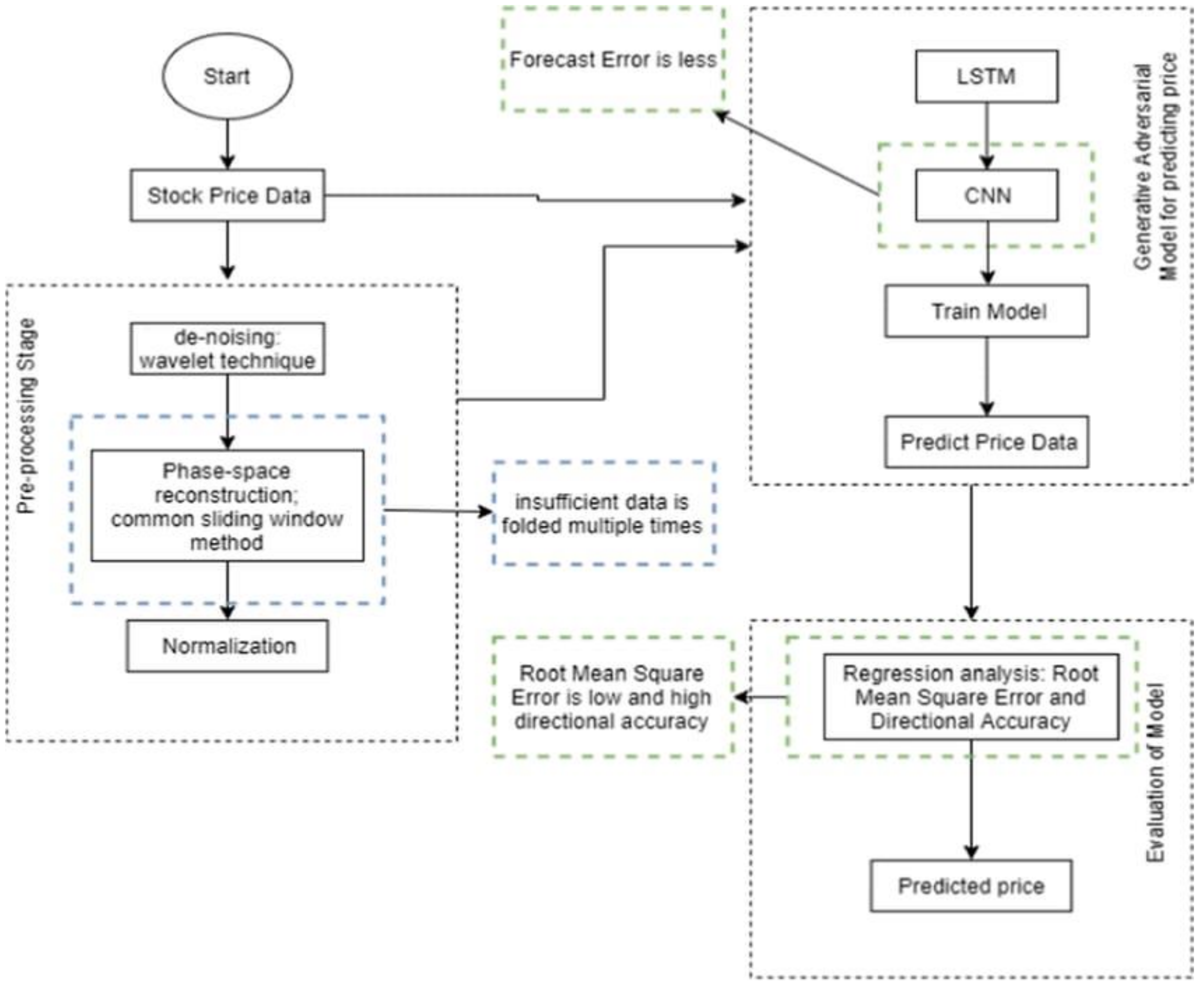
7.**Research and Analysis:** To conduct financial research, test hypotheses, and gain insights into market behaviour, which can contribute to academic research, industry reports, and investment recommendations.

It's important to note that while stock price prediction can be a valuable tool in financial decision-making, it also comes with inherent uncertainties and risks. Predictions should be used in conjunction with other forms of analysis and should not be the sole basis for investment decisions.

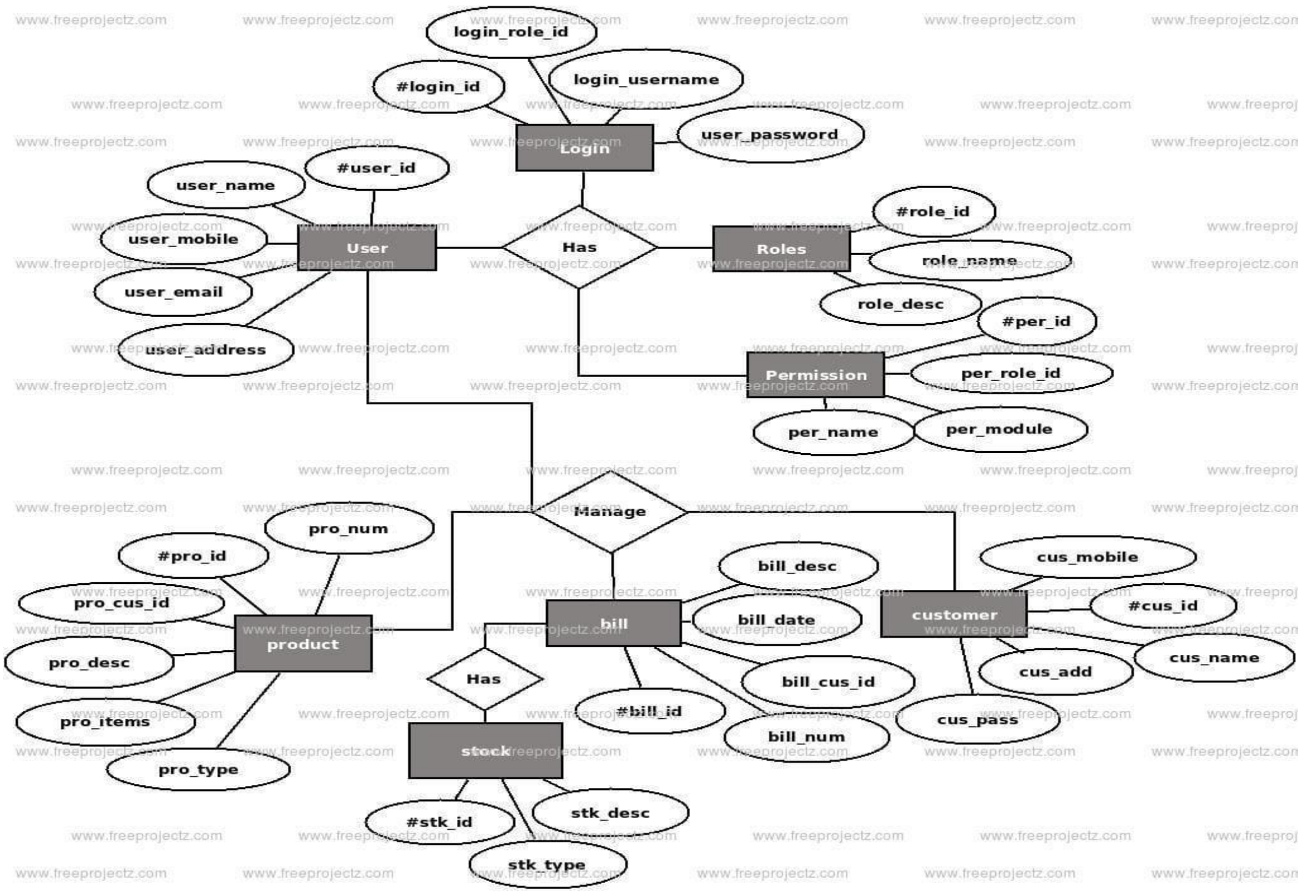
## DESIGN THINKING:

## SYSTEM ARCHITECTURE:

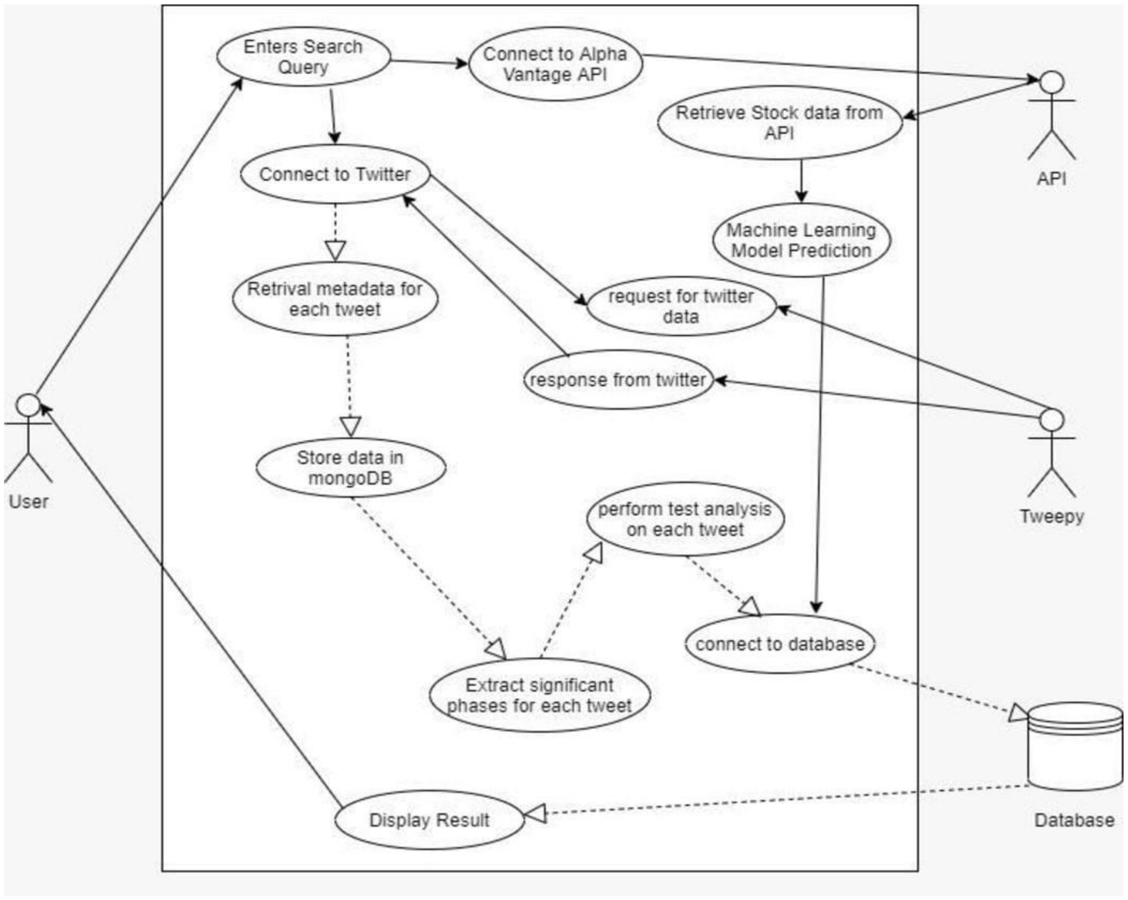
System architecture refers to the organized structure and arrangement of components, including data sources, predictive models, data preprocessing, and user interfaces, which together enable the process of collecting, processing, and delivering stock price forecasts to users efficiently and accurately.


Start
Stock Price Data
Pre-processing Stage
de-noising: wavelet technique
Phase-space reconstruction; common sliding window method
Normalization
insufficient data is folded multiple times
Forecast Error is less
LSTM
CNN
Train Model
Predict Price Data
Generative Adversarial Model for predicting price
Root Mean Square Error is low and high directional accuracy
Regression analysis: Root Mean Square Error and Directional Accuracy
Predicted price
Evaluation of Model

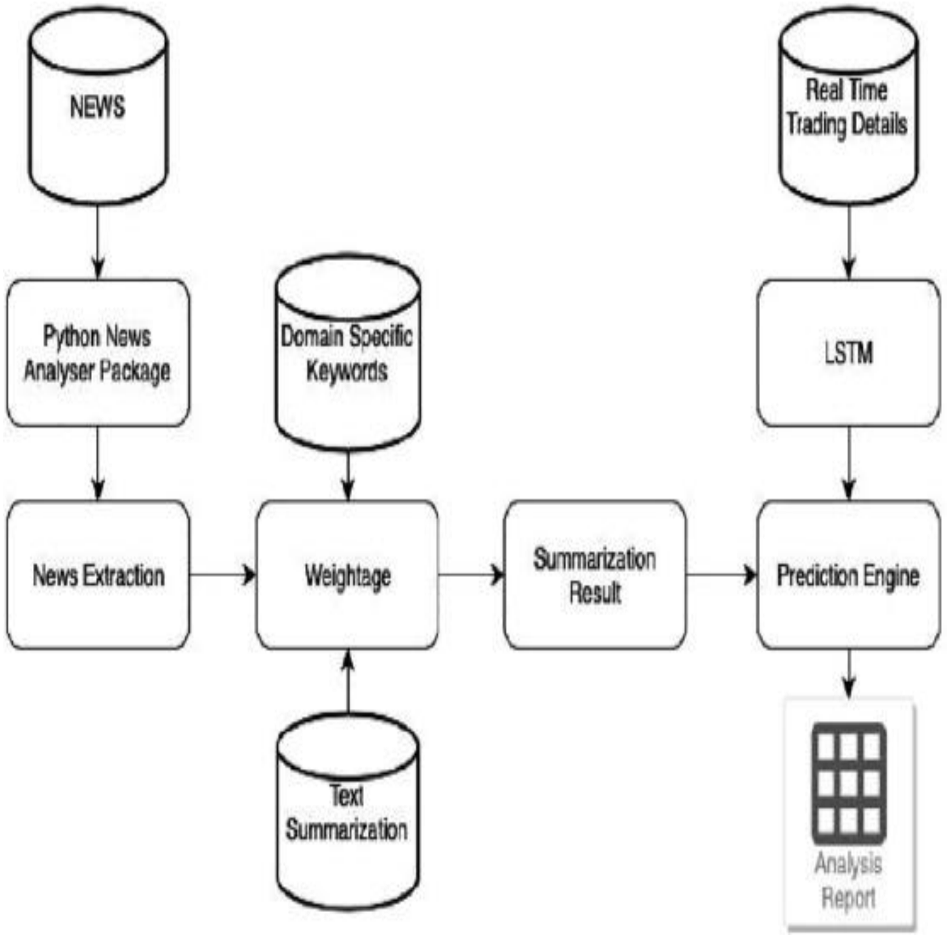
## ER DIAGRAM:


login_role_id
#login_id
login_username
user_password
Login
#user_id
user_name
user_mobile
User
user_email
user_address
Has
Roles
#role_id
role_name
role_desc
Permission
#per_id
per_role_id
per_name
per_module
Manage
pro_num
#pro_id
pro_cus_id
product
pro_desc
pro_items
pro_type
bill
bill_desc
bill_date
bill_cus_id
#bill_id
bill_num
Has
stock
#stk_id
stk_desc
stk_type
customer
cus_mobile
#cus_id
cus_name
cus_add
cus_pass

## USE CASE DIAGRAM:


Enters Search Query
Connect to Alpha Vantage API
Retrieve Stock data from API
API
Connect to Twitter
Machine Learning Model Prediction
Retrival metadata for each tweet
request for twitter data
response from twitter
User
Store data in mongoDB
Tweepy
perform test analysis on each tweet
connect to database
Extract significant phases for each tweet
Display Result
Database

## BLOCK DIAGRAM:

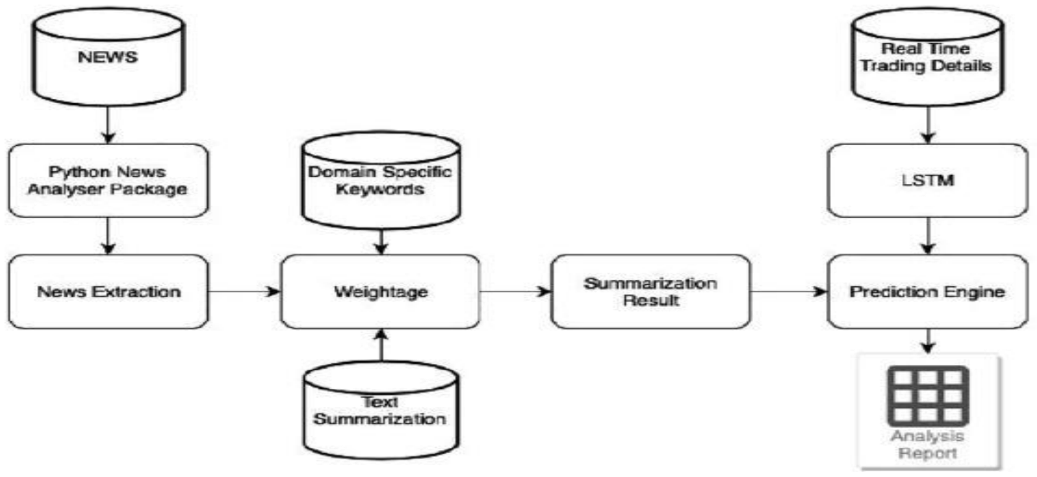


# CHAPTER 2

## INNOVATION:

Innovation in stock price prediction refers to the development and application of new techniques, technologies, and methodologies to improve the accuracy, speed, and reliability of forecasting future stock prices. These innovations aim to address the inherent challenges and complexities of financial markets, helping investors, traders, and financial institutions make more informed decisions.
Innovations in stock price prediction aim to address the challenges and complexities associated with forecasting stock prices. Here are some innovative approaches and technologies that can help solve the problem:

## DESIGN:



## DEEP LEARNING:

Deep learning is a subset of machine learning that involves the use of artificial neural networks, commonly referred to as neural networks, to model and solve complex problems.
In the context of stock price prediction, deep learning and neural networks offer innovative techniques for analysing historical stock price data and making forecasts.

## → USING PANDAS:

Pandas is a powerful Python library used for data manipulation and analysis. It provides data structures like data frames, which are well-suited for handling tabular data, such as historical stock price datasets. Pandas simplifies various aspects of data manipulation:

1.**Data Loading:**

- Pandas provides functions like **read_csv()** or **read_excel()** to load data from external sources into data frames, which are Pandas' primary data structure for tabular data.

2.**Data Cleaning:**

- Pandas offers various methods to clean and preprocess data, including functions like **dropna(), fillna(),** and **drop_duplicates()** for handling missing values, and duplicates, respectively.

3**. Indexing and Slicing**:

- You can use Pandas' powerful indexing and slicing capabilities to select specific rows and columns based on conditions. For instance, you can filter data for a specific date range or select columns of interest.

4**. Grouping and Aggregation**:

- Pandas provides functions like **groupby()** and **agg()** to group data based on specific attributes (e.g., stock symbols) and perform aggregation operations (e.g., mean, sum) within each group.

5.**Resampling Time Series Data:**

- When dealing with time series data, Pandas' resample() method allows you to change the frequency of data (e.g., daily to monthly) and apply aggregation functions to the resampled data.

6.**Merging and Joining Data:**

- You can use Pandas' merge() and join() functions to combine data from different sources or data frames based on common columns or indices.

7.**Data Visualization Integration:**

- Pandas seamlessly integrates with data visualization libraries like Matplotlib and Seaborn, allowing you to create informative plots and charts to visualize stock price trends.

## →USING NUMPY:

NumPy (Numerical Python) is a fundamental Python library for numerical computing that provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays efficiently. In stock price prediction, NumPy is valuable for various tasks:

1.**Numerical Operations:**

- NumPy is fundamental for numerical operations in Python. You can use NumPy arrays to perform element-wise mathematical operations on data, such as computing logarithms or exponentials.

2.**Array Manipulation**:

- NumPy provides extensive array manipulation functions, including reshaping arrays, stacking arrays, and transposing arrays. These can be handy for preparing data for analysis.

3.**Handling Missing Values**:

- NumPy arrays can be used to identify and handle missing values by applying logical conditions (e.g., np.isnan()) or filling missing values with specific values (e.g., np.nan_to_num()).

4.**Statistical Analysis**:

- NumPy includes a wide range of statistical functions to compute measures like mean, median, standard deviation, and correlation coefficients, which are useful for analyzing stock price data.

5.**Random Number Generation**:

- NumPy's random module can generate random numbers and random samples. This is valuable for simulating stock price scenarios or generating random portfolios for risk analysis.

Pandas simplifies the process of data manipulation, making it a vital tool for preparing and transforming stock data for predictive modelling and analysis in the financial industry. It enhances the efficiency and effectiveness of data preparation tasks, enabling better-informed decisions in stock price prediction.

NumPy is a foundational library that enhances the efficiency and flexibility of numerical computations in Python. It is a critical tool for handling and manipulating numerical data in stock price prediction tasks, where efficient mathematical operations and data transformations are essential for accurate modelling and analysis.

Both Pandas and NumPy are highly versatile libraries that complement each other in data manipulation tasks. Pandas excels at handling structured data, while NumPy provides the numerical backbone for mathematical and statistical operations. Together, they form a powerful toolkit for data preprocessing and analysis in stock price prediction.

**Stock price prediction** is a challenging task due to its inherent complexity and numerous factors that influence stock prices. While no algorithm can guarantee accurate predictions, here's a commonly used algorithmic approach for stock price prediction:

## Long Short-Term Memory (LSTM) Neural Networks:

LSTM is a type of recurrent neural network (RNN) designed to model sequences and time series data, making it well-suited for stock price prediction. Here's how LSTM can be applied:

1.**Data Preparation**:

- Gather historical stock price data, including daily or intraday open, high, low, and close prices, as well as trading volumes.
- Create a dataset with sequential input features (e.g., past stock prices) and a target variable (e.g., future stock prices or returns).

2.**Data Preprocessing**:

- Normalize or standardize the data: Scaling input features to a consistent range (e.g., between 0 and 1) can help improve the convergence of the LSTM model during training.
- Split the data into training, validation, and test sets. A common split might be 70% for training, 15% for validation (used for model tuning), and 15% for testing (used to evaluate the final model).

3.**Model Architecture**:

- Design an LSTM neural network with one or more layers of LSTM cells. You can experiment with the network architecture by adjusting the number of layers, units in each layer, and dropout layers to prevent overfitting.
- Input Layer: Accepts sequential data with multiple features, often shaped as (batch_size, time_steps, num_features).
- LSTM Layers: Stacked LSTM layers to capture temporal dependencies. The number of layers and units in each layer can vary.
- Output Layer: Typically a dense layer with one neuron for regression tasks (predicting stock prices) or multiple neurons for classification tasks (e.g., predicting buy/sell signals).

4.**Training**:

- Train the LSTM model using the training dataset. The model learns to capture temporal patterns and dependencies in the data. Define loss function: Choose an appropriate loss function (e.g., Mean Squared Error for regression).

- Batch Training: Train the model in batches of data rather than using the entire dataset at once. This helps with memory efficiency and convergence.
- Implement early stopping and hyperparameter tuning to optimize the model's performance.

5.**Validation and Evaluation**:

- Evaluate the model on the validation dataset using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or others relevant to your specific goals.
- Ensure that the model's performance on the validation set is satisfactory and not showing signs of overfitting.

6.**Testing**:

- Once the model is trained and validated, use it to make predictions on unseen data from the test set.
- Evaluate the model's accuracy and reliability on the test data.

7.**Deployment and Monitoring:**

- Deploy the trained model in a real-time or batch prediction system, where it can continuously make predictions or generate signals for trading strategies.
- Continuously monitor the model's performance and retrain it periodically with new data to adapt to changing market condition.

Thus, LSTM models can capture temporal patterns in stock price data, they are not guaranteed to provide accurate predictions. Stock prices are influenced by a multitude of factors, including market sentiment, economic events, and news, which are often challenging to capture fully with historical price data alone.

Therefore, combining LSTM models with comprehensive market analysis and risk management strategies is crucial for informed investment decisions.

It's important to note that stock price prediction is highly complex and subject to various external factors, including news events, economic indicators, and market sentiment. Therefore, combining machine learning models like LSTM with comprehensive market analysis and risk management strategies is essential for informed investment decisions.

It requires substantial labelled data for effective training and has led to breakthroughs in computer vision, language translation, and autonomous systems. Despite its successes, deep learning also poses challenges related to data availability, model complexity, and interpretability.

Deep learning is a powerful and transformative subset of machine learning that leverages deep neural networks to learn complex patterns and representations from data. Its impact extends across various industries and research areas, offering the potential to solve intricate problems and drive technological advancements.

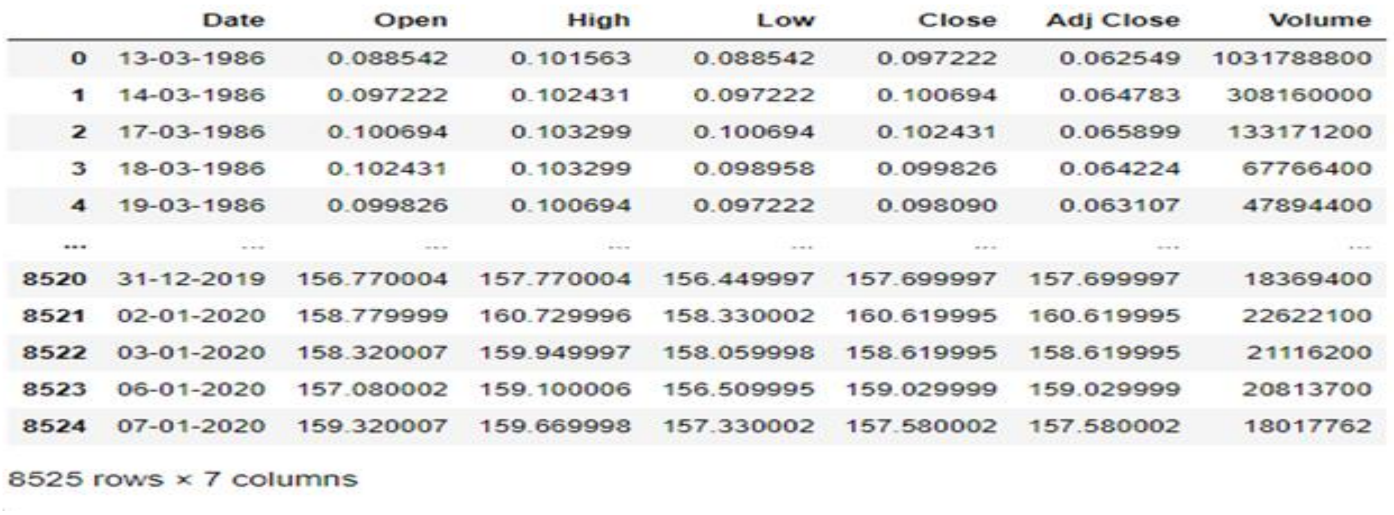
# CHAPTER 3

## BUILDING THE PROJECT:

Stock Price Prediction using machine learning is the process of predicting the future value of a stock traded on a stock exchange for reaping profits. With multiple factors involved in predicting stock prices, it is challenging to predict stock prices with high accuracy, and this is where machine learning plays a vital role.
Stock Price Prediction System is a data-driven project aimed at forecasting the future prices of publicly traded stocks. This project leverages advanced machine learning and data analysis techniques to provide insights into stock market behaviour and assist investors, traders, and financial decisions.
This system incorporates several critical components to achieve its objectives. It begins with data collection and preprocessing, where historical stock price data, trading volumes, and external factors such as economic indicators and news sentiment are meticulously curated and cleaned.

## GIVEN DATASET:

LINK:
https//www.kaggle.com/datasets/prasoonkottrathil/microsoft-lifetime-stocks-datasets



| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| 0 | 13-03-1986 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.062549 | 1031788800 |
| 1 | 14-03-1986 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.064783 | 308160000 |
| 2 | 17-03-1986 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.065899 | 133171200 |
| 3 | 18-03-1986 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.064224 | 67766400 |
| 4 | 19-03-1986 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.063107 | 47894400 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 8520 | 31-12-2019 | 156.770004 | 157.770004 | 156.449997 | 157.699997 | 157.699997 | 18369400 |
| 8521 | 02-01-2020 | 158.779999 | 160.729996 | 158.330002 | 160.619995 | 160.619995 | 22622100 |
| 8522 | 03-01-2020 | 158.320007 | 159.949997 | 158.059998 | 158.619995 | 158.619995 | 21116200 |
| 8523 | 06-01-2020 | 157.080002 | 159.100006 | 156.509995 | 159.029999 | 159.029999 | 20813700 |
| 8524 | 07-01-2020 | 159.320007 | 159.669998 | 157.330002 | 157.580002 | 157.580002 | 18017762 |

8525 rows × 7 columns

## PREPROCESSING THE DATA:

Preprocessing the dataset in stock price prediction involves cleaning, normalizing, and transforming the data for better analysis. It helps remove noise and inconsistencies to improve the accuracy of the prediction models.

- **Data Loading:** Begin by loading the dataset into your environment. Depending on the format of the data (e.g., CSV, Excel, SQL database), you'll use different libraries or methods to import the data into a data structure like a DataFrame.
- **Data Collection:** Gather historical stock prices, trading volumes, and relevant financial data. Sources could include financial databases, APIs, or web scraping tools.

**Data Cleaning:**

1.Handle Missing Values: Check for missing data and decide how to handle it. You can choose to remove rows with missing values, fill them with the mean, median, or a specific value, or use more advanced imputation methods

2.Data Validation: Check for data inconsistencies or errors and correct them. This may involve handling outliers or anomalies in the data.

- **Data Pre-processing:** Clean the data, handle missing values, and perform feature engineering. This step might involve normalization, scaling, or transforming the data to make it suitable for the chosen model.Data preprocessing may involve more or fewer steps depending on your dataset and objectives.
- **Data Transformation:**

1.Feature Selection: Choose relevant features that might affect stock prices, such as historical prices, trading volumes, news sentiment, economic indicators, and company-specific information

2.Model Selection: Select an appropriate machine learning algorithm such as linear regression, decision trees, random forests, or deep learning models like recurrent neural networks (RNNs) or long short-term memory networks (LSTMs)

- Training the Model: Use a portion of the data to train the model, adjusting parameters and hyper parameters to optimize performance. Training a machine learning model involves selecting an appropriate algorithm, preparing your data, and using that data to train the model.
- Model Evaluation: Assess the model's performance using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE) on a validation dataset. Model evaluation is a critical step in the development and assessment of machine learning models. It involves assessing the model's performance to determine how well it generalizes to new, unseen data.
- Testing the Model: Apply the trained model to a separate test dataset to evaluate its predictive power. Test models are simplified representations of real-world systems used for experimentation and analysis.They help assess the behavior and performance of systems under various conditions, providing insights and aiding decision-making.
- Iterate and Refine: Fine-tune the model by iterating on the pre-processing steps, feature selection, and model selection to improve predictive accuracy. The process begins with an initial version, idea, or solution. Instead of seeking perfection from the start, you make incremental improvements through successive cycles or iterations. Each iteration allows for feedback, learning, and adaptation.

It’s important to note that stock market prediction is inherently complex and subject to a variety of external factors, including market sentiment, geopolitical events, and economic indicators, which might not be fully captured by historical data alone. Therefore, it's essential to exercise caution and understand the limitations of any predictive mode

## <u>IMPORTANCE OF LOADING AND PREPROCESSING THE DATASET</u>:

Loading and preprocessing the dataset is an important first step in building any machine learning model. However, it is especially important for house price prediction models, as house price datasets are often complex and noisy.

By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately

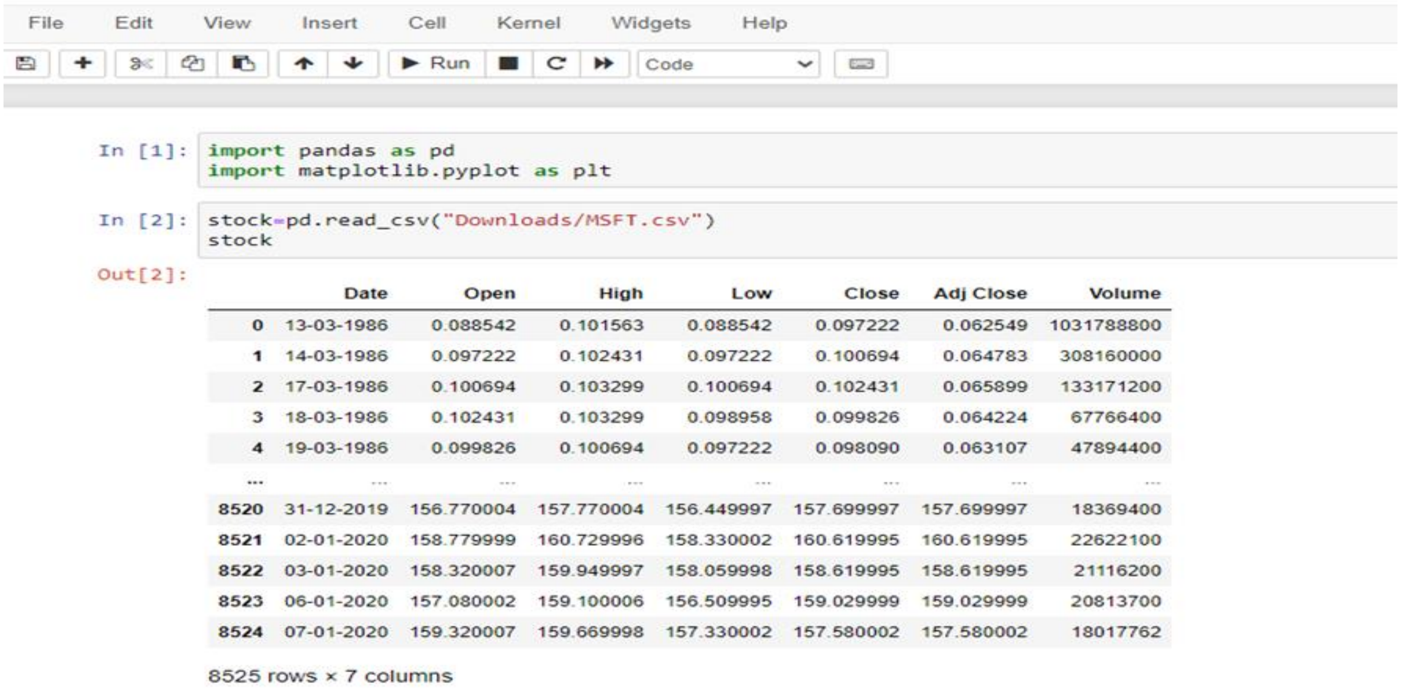
## NECESSARY STEPS TO FOLLOW:

1.**Import Libraries:**
Start by importing the necessary libraries:
<u>Code:</u>

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

**2. Load the Dataset:**
Load your dataset into a Pandas DataFrame. You can typically find stock price datasets in CSV format, but you can adapt this code to other formats as needed.
<u>Code:</u>

```
df = pd.read_csv(“D:downloads/MFST.csv”)
```



```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt

In [2]: stock=pd.read_csv("Downloads/MSFT.csv")
        stock

Out[2]:
            Date        Open        High         Low       Close   Adj Close      Volume
   0  13-03-1986    0.088542    0.101563    0.088542    0.097222    0.062549  1031788800
   1  14-03-1986    0.097222    0.102431    0.097222    0.100694    0.064783   308160000
   2  17-03-1986    0.100694    0.103299    0.100694    0.102431    0.065899   133171200
   3  18-03-1986    0.102431    0.103299    0.098958    0.099826    0.064224    67766400
   4  19-03-1986    0.099826    0.100694    0.097222    0.098090    0.063107    47894400
 ...         ...         ...         ...         ...         ...         ...         ...
8520  31-12-2019  156.770004  157.770004  156.449997  157.699997  157.699997    18369400
8521  02-01-2020  158.779999  160.729996  158.330002  160.619995  160.619995    22622100
8522  03-01-2020  158.320007  159.949997  158.059998  158.619995  158.619995    21116200
8523  06-01-2020  157.080002  159.100006  156.509995  159.029999  159.029999    20813700
8524  07-01-2020  159.320007  159.669998  157.330002  157.580002  157.580002    18017762

8525 rows × 7 columns
```

**3. Exploratory Data Analysis (EDA):** Perform EDA to understand your data better. This includes checking for missing values, exploring the data ' s statistics, and visualizing it to identify patterns.
**<u>Code:</u>**

```
# Check for missing values
   print(df.isnull().sum())
# Explore statistics
    print(df.describe())
 # Visualize the data (e.g., histograms, scatter plots, etc.)
```

**4. Split the Data:** Split your dataset into training and testing sets. This helps you evaluate your model' s performance later. Thus it can be implemented by using the dataset and extracted for splitting the dataset.

**Code:**

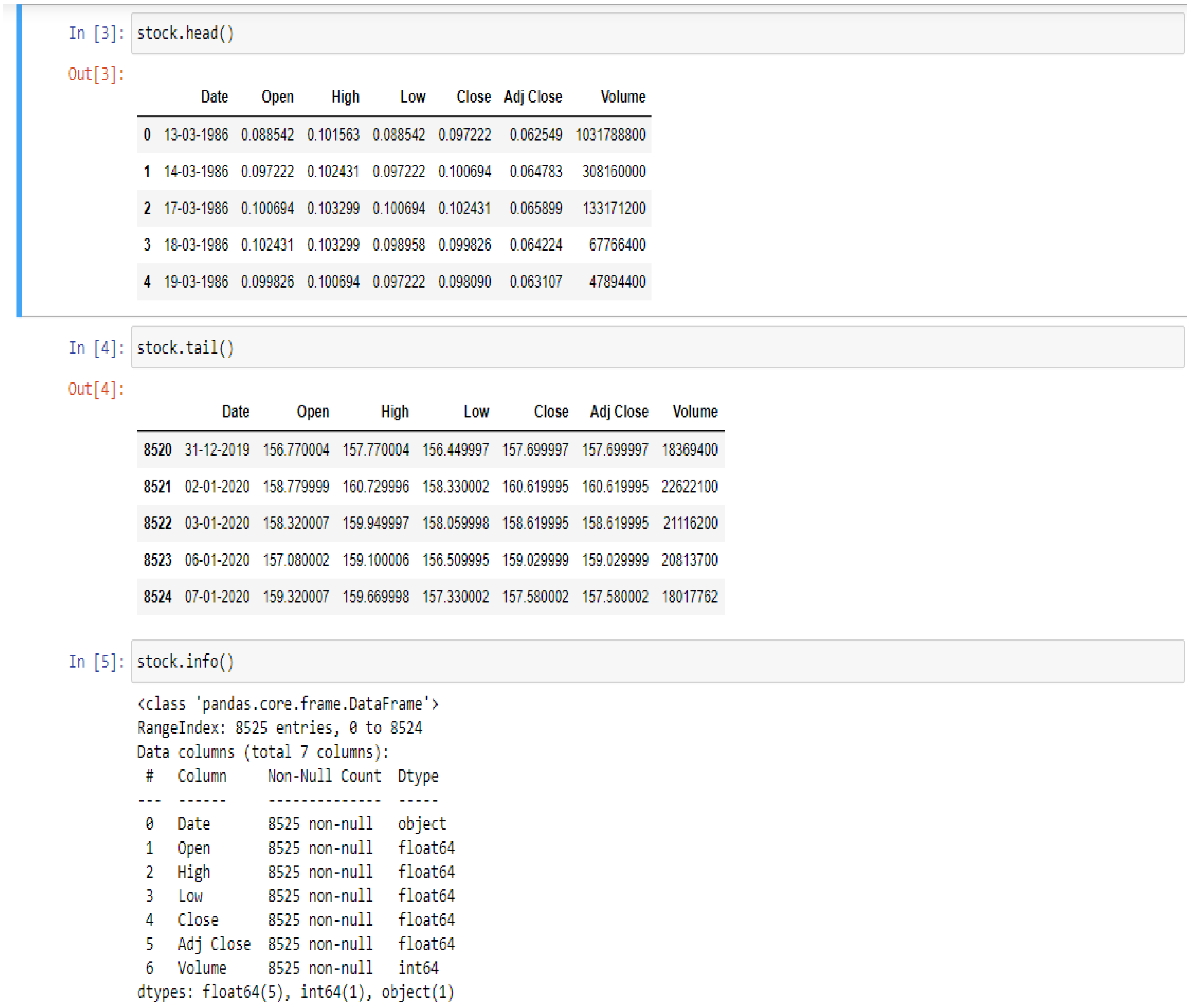
```
X = df.drop(' low ' , axis=1) # Features
y = df[' price '] # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

**5.DISPLAYING THE PARTICULAR ROW:**

It can be implemented by performing the particular rows in the dataset can be displayed.

Code:

```
stock.head()
stock.tail()
stock.info()
```



```
In [3]: stock.head()
Out[3]:
```

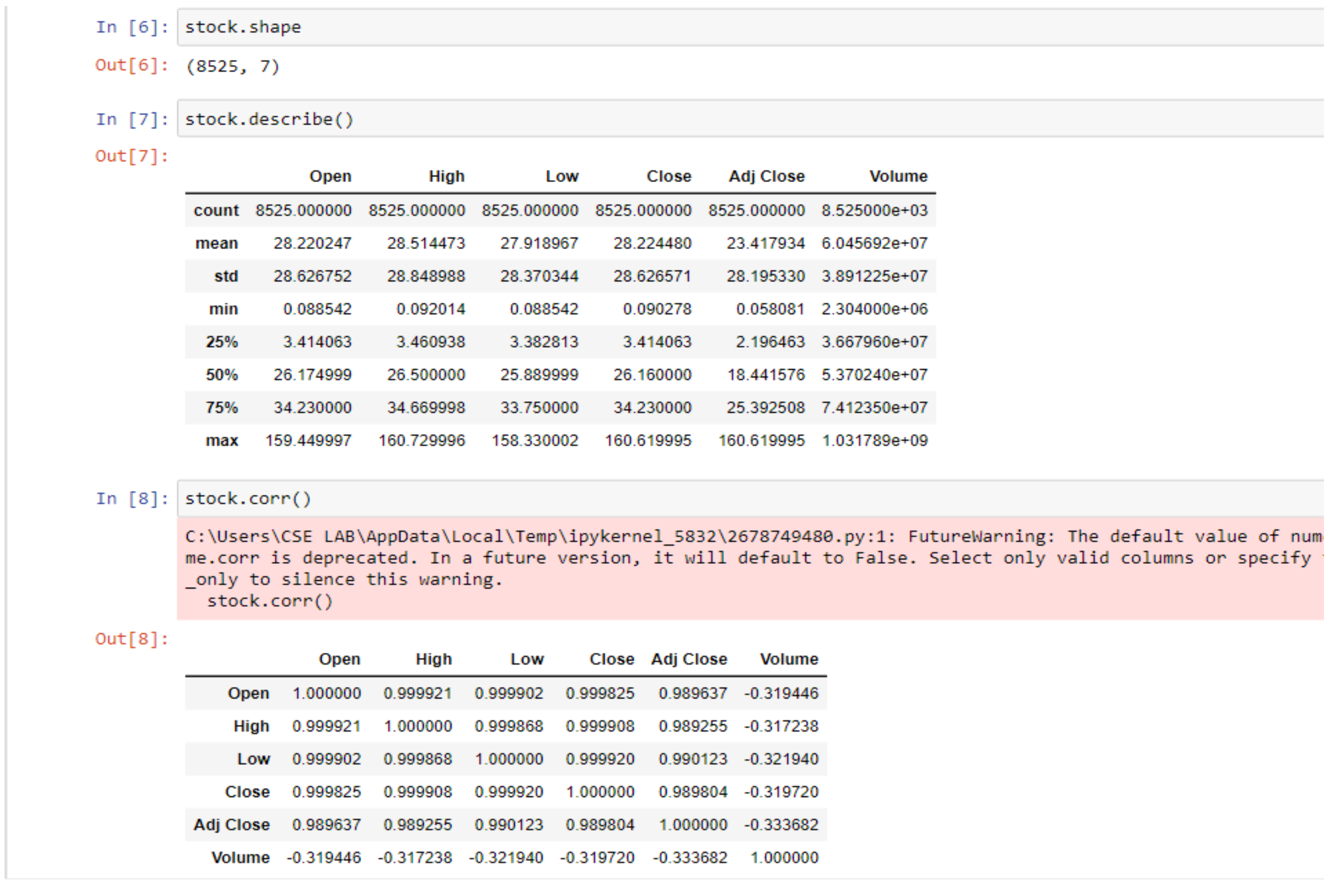
| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| **0** | 13-03-1986 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.062549 | 1031788800 |
| **1** | 14-03-1986 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.064783 | 308160000 |
| **2** | 17-03-1986 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.065899 | 133171200 |
| **3** | 18-03-1986 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.064224 | 67766400 |
| **4** | 19-03-1986 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.063107 | 47894400 |

```
In [4]: stock.tail()
Out[4]:
```

| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| **8520** | 31-12-2019 | 156.770004 | 157.770004 | 156.449997 | 157.699997 | 157.699997 | 18369400 |
| **8521** | 02-01-2020 | 158.779999 | 160.729996 | 158.330002 | 160.619995 | 160.619995 | 22622100 |
| **8522** | 03-01-2020 | 158.320007 | 159.949997 | 158.059998 | 158.619995 | 158.619995 | 21116200 |
| **8523** | 06-01-2020 | 157.080002 | 159.100006 | 156.509995 | 159.029999 | 159.029999 | 20813700 |
| **8524** | 07-01-2020 | 159.320007 | 159.669998 | 157.330002 | 157.580002 | 157.580002 | 18017762 |

```
In [5]: stock.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8525 entries, 0 to 8524
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       8525 non-null   object 
 1   Open       8525 non-null   float64
 2   High       8525 non-null   float64
 3   Low        8525 non-null   float64
 4   Close      8525 non-null   float64
 5   Adj Close  8525 non-null   float64
 6   Volume     8525 non-null   int64  
dtypes: float64(5), int64(1), object(1)
```

## 6.TO VIEW SHAPE,CORR:



```
In [6]: stock.shape
Out[6]: (8525, 7)
```

```
In [7]: stock.describe()
Out[7]:
```

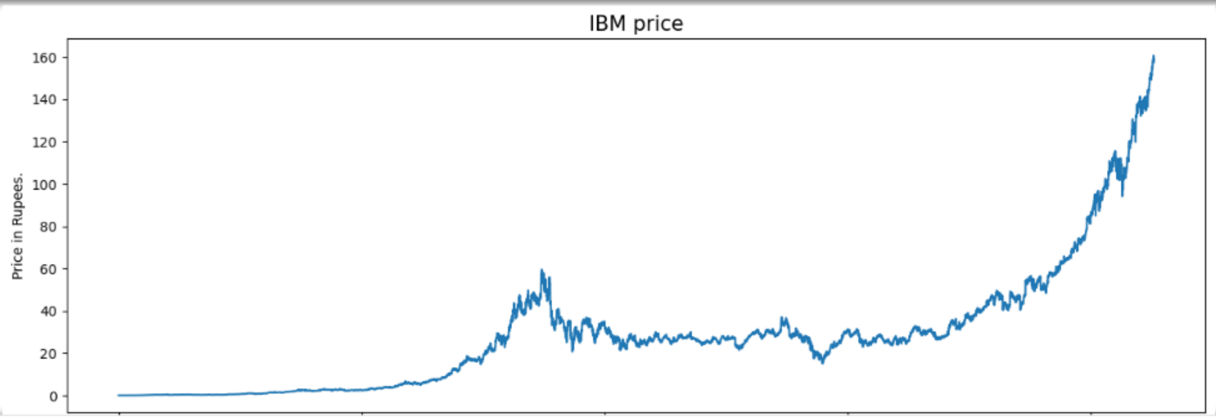
| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **count** | 8525.000000 | 8525.000000 | 8525.000000 | 8525.000000 | 8525.000000 | 8.525000e+03 |
| **mean** | 28.220247 | 28.514473 | 27.918967 | 28.224480 | 23.417934 | 6.045692e+07 |
| **std** | 28.626752 | 28.848988 | 28.370344 | 28.626571 | 28.195330 | 3.891225e+07 |
| **min** | 0.088542 | 0.092014 | 0.088542 | 0.090278 | 0.058081 | 2.304000e+06 |
| **25%** | 3.414063 | 3.460938 | 3.382813 | 3.414063 | 2.196463 | 3.667960e+07 |
| **50%** | 26.174999 | 26.500000 | 25.889999 | 26.160000 | 18.441576 | 5.370240e+07 |
| **75%** | 34.230000 | 34.669998 | 33.750000 | 34.230000 | 25.392508 | 7.412350e+07 |
| **max** | 159.449997 | 160.729996 | 158.330002 | 160.619995 | 160.619995 | 1.031789e+09 |

```
In [8]: stock.corr()
        C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\2678749480.py:1: FutureWarning: The default value of num
        me.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify
        _only to silence this warning.
          stock.corr()
Out[8]:
```

| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| **Open** | 1.000000 | 0.999921 | 0.999902 | 0.999825 | 0.989637 | -0.319446 |
| **High** | 0.999921 | 1.000000 | 0.999868 | 0.999908 | 0.989255 | -0.317238 |
| **Low** | 0.999902 | 0.999868 | 1.000000 | 0.999920 | 0.990123 | -0.321940 |
| **Close** | 0.999825 | 0.999908 | 0.999920 | 1.000000 | 0.989804 | -0.319720 |
| **Adj Close** | 0.989637 | 0.989255 | 0.990123 | 0.989804 | 1.000000 | -0.333682 |
| **Volume** | -0.319446 | -0.317238 | -0.321940 | -0.319720 | -0.333682 | 1.000000 |

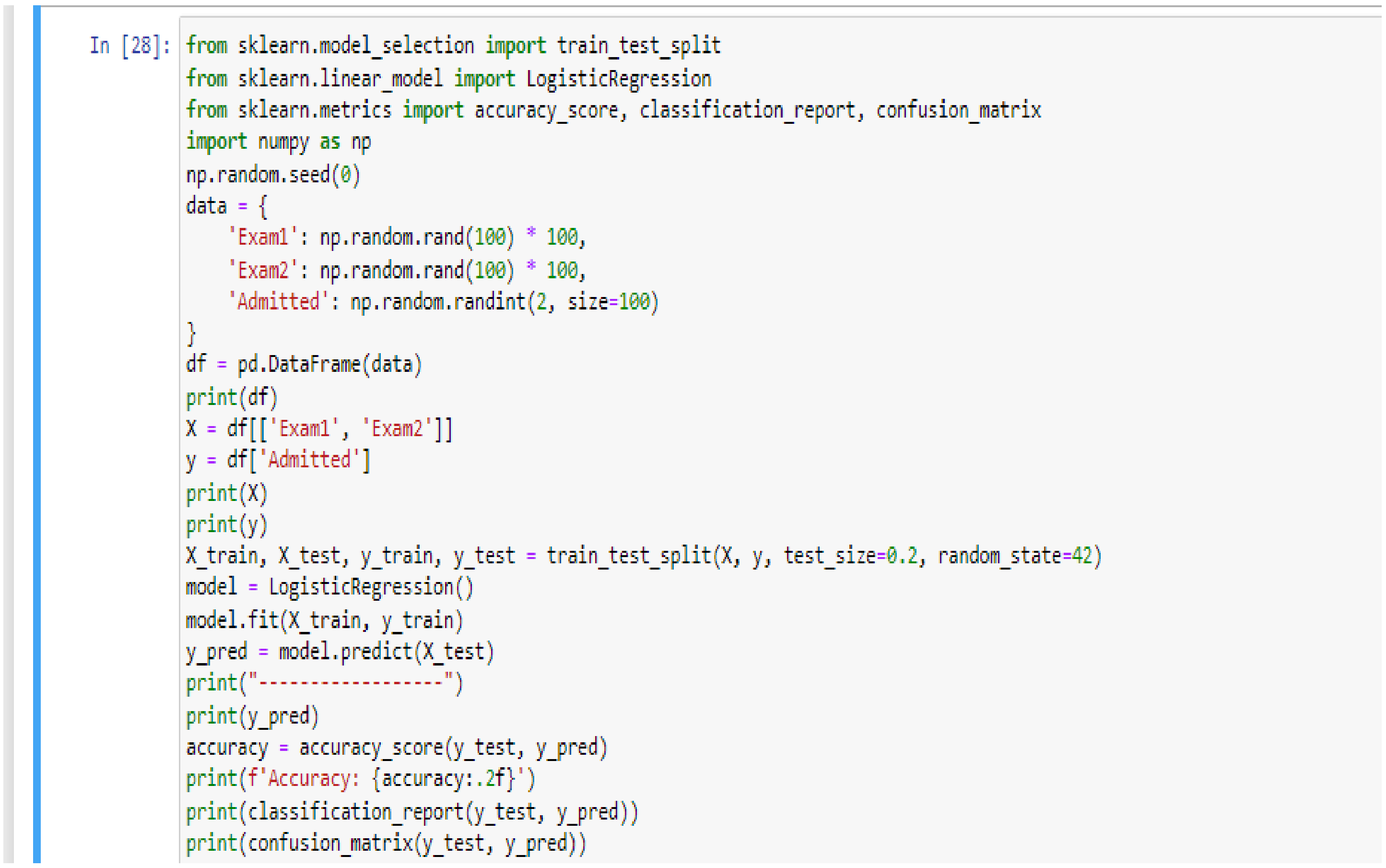
## 7. CALCUALTING THE PRICE AND SUM:

```
In [12]: plt.figure(figsize=(15,5))
         plt.plot(stock['Close'])
         plt.title('IBM price', fontsize=15)
         plt.ylabel('Price in Rupees.')
         plt.show()
```

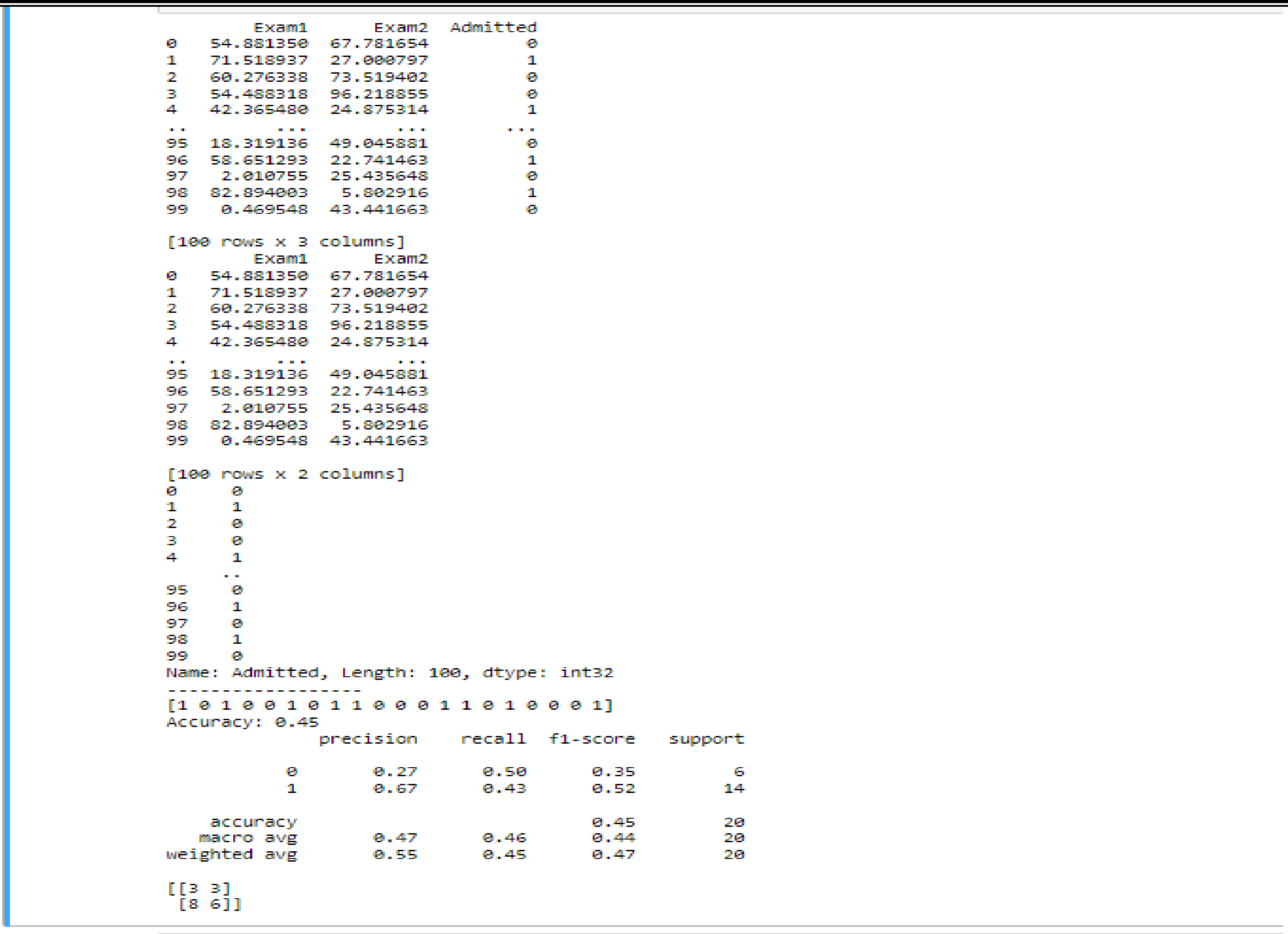


```
In [13]: stock.isnull().sum()

Out[13]: Date         0
         Open         0
         High         0
         Low          0
         Close        0
         Adj Close    0
         Volume       0
         dtype: int64
```

## 8. REGRESSION AND ACCURACY:

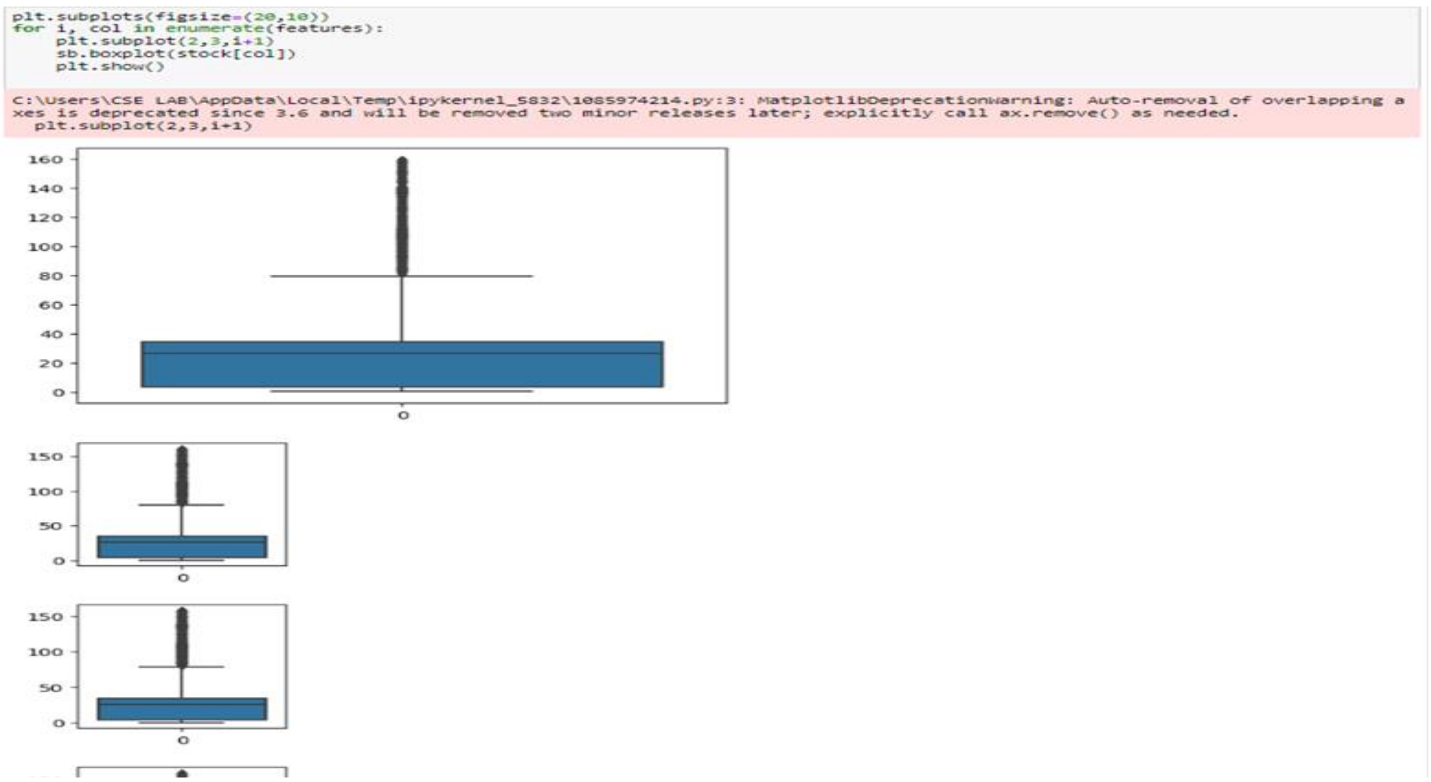


```
In [28]: from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
         import numpy as np
         np.random.seed(0)
         data = {
             'Exam1': np.random.rand(100) * 100,
             'Exam2': np.random.rand(100) * 100,
             'Admitted': np.random.randint(2, size=100)
         }
         df = pd.DataFrame(data)
         print(df)
         X = df[['Exam1', 'Exam2']]
         y = df['Admitted']
         print(X)
         print(y)
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
         model = LogisticRegression()
         model.fit(X_train, y_train)
         y_pred = model.predict(X_test)
         print("-----------------")
         print(y_pred)
         accuracy = accuracy_score(y_test, y_pred)
         print(f'Accuracy: {accuracy:.2f}')
         print(classification_report(y_test, y_pred))
         print(confusion_matrix(y_test, y_pred))
```



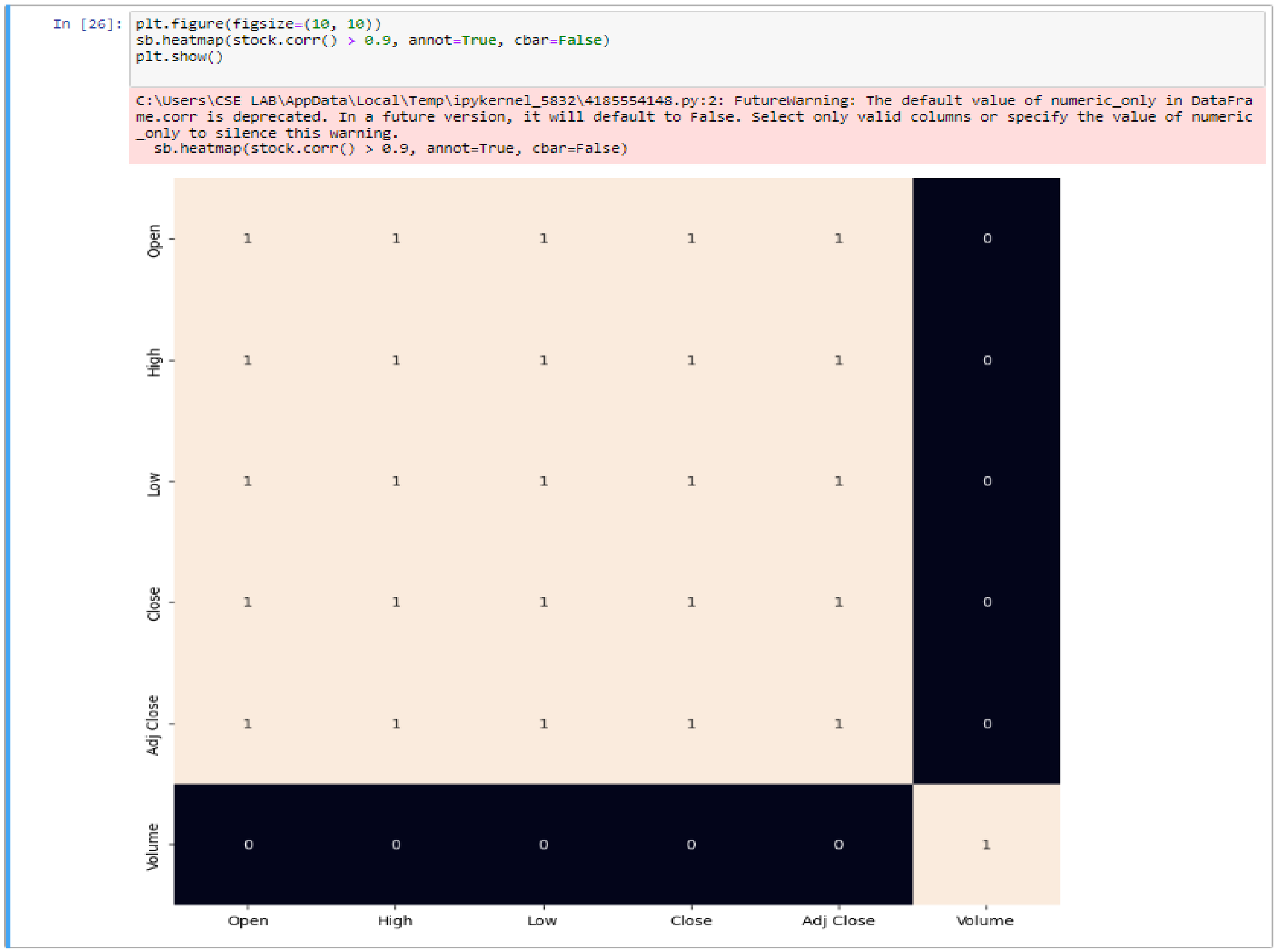
```
       Exam1      Exam2  Admitted
0   54.881350  67.781654         0
1   71.518937  27.000797         1
2   60.276338  73.519402         0
3   54.488318  96.218855         0
4   42.365480  24.875314         1
..        ...        ...       ...
95  18.319136  49.045881         0
96  58.651293  22.741463         1
97   2.010755  25.435648         0
98  82.894003   5.802916         1
99   0.469548  43.441663         0

[100 rows x 3 columns]
       Exam1      Exam2
0   54.881350  67.781654
1   71.518937  27.000797
2   60.276338  73.519402
3   54.488318  96.218855
4   42.365480  24.875314
..        ...        ...
95  18.319136  49.045881
96  58.651293  22.741463
97   2.010755  25.435648
98  82.894003   5.802916
99   0.469548  43.441663

[100 rows x 2 columns]
0     0
1     1
2     0
3     0
4     1
     ..
95    0
96    1
97    0
98    1
99    0
Name: Admitted, Length: 100, dtype: int32
------------------
[1 0 1 0 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1]
Accuracy: 0.45
              precision    recall  f1-score   support

           0       0.27      0.50      0.35         6
           1       0.67      0.43      0.52        14

    accuracy                           0.45        20
   macro avg       0.47      0.46      0.44        20
weighted avg       0.55      0.45      0.47        20

[[3 3]
 [8 6]]
```

## 9.VIEW AS CHART:



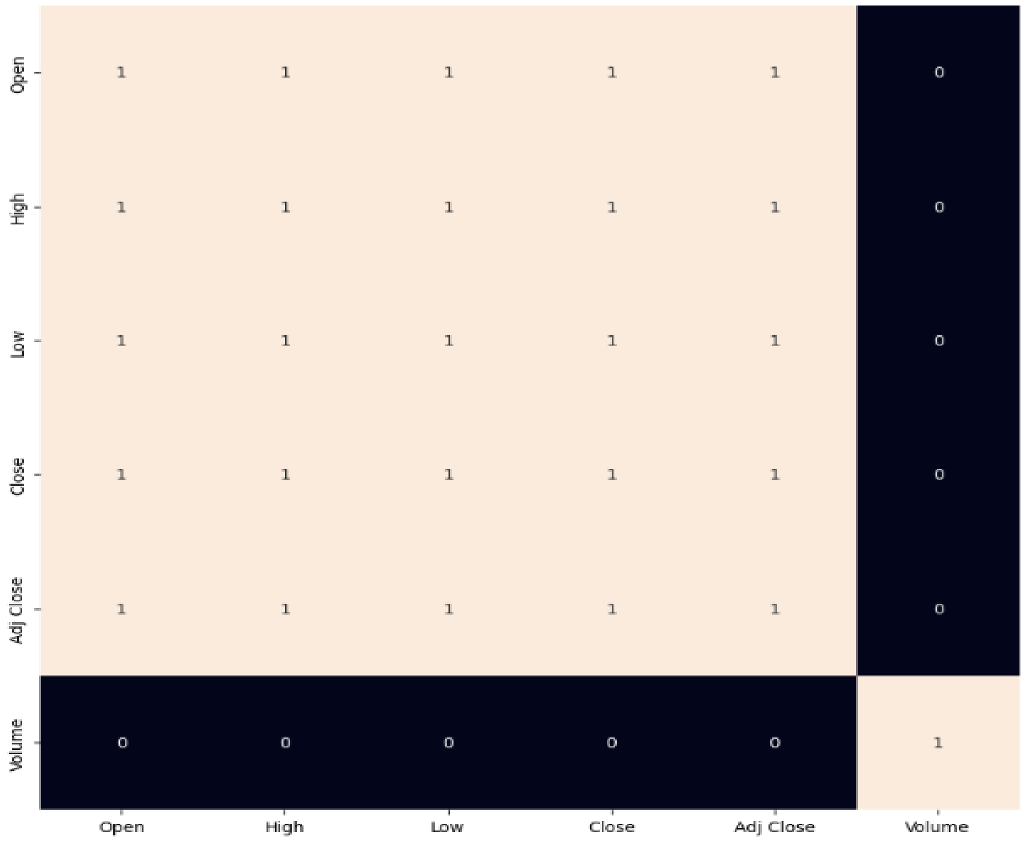
```
plt.subplots(figsize=(20,10))
for i, col in enumerate(features):
    plt.subplot(2,3,i+1)
    sb.boxplot(stock[col])
    plt.show()
```

```
C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\1085974214.py:3: MatplotlibDeprecationWarning: Auto-removal of overlapping a
xes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() as needed.
  plt.subplot(2,3,i+1)
```



```
In [26]: plt.figure(figsize=(10, 10))
         sb.heatmap(stock.corr() > 0.9, annot=True, cbar=False)
         plt.show()
```

```
C:\Users\CSE LAB\AppData\Local\Temp\ipykernel_5832\4185554148.py:2: FutureWarning: The default value of numeric_only in DataFra
me.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric
_only to silence this warning.
  sb.heatmap(stock.corr() > 0.9, annot=True, cbar=False)
```



|  | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| Open | 1 | 1 | 1 | 1 | 1 | 0 |
| High | 1 | 1 | 1 | 1 | 1 | 0 |
| Low | 1 | 1 | 1 | 1 | 1 | 0 |
| Close | 1 | 1 | 1 | 1 | 1 | 0 |
| Adj Close | 1 | 1 | 1 | 1 | 1 | 0 |
| Volume | 0 | 0 | 0 | 0 | 0 | 1 |

# CHAPTER 4

## OVERVIEW:

The following is an overview of the process of building a stock price prediction model by feature selection, model training, and evaluation.

### 1. Prepare the data:

This includes cleaning the data, removing outliers, and handling missing values.

### 2. Perform feature selection:

This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

### 3.Train the model:

There are many different machine learning algorithms that can be usedfor house price prediction. Some popular choices include linear regression, random forests, and gradient boosting machines. Training a model" is a fundamental concept in machine learning and deep learning. It refers to the process of teaching a machine learning algorithm or neural network to make predictions or classifications based on input data.

### 4. Evaluate the model:

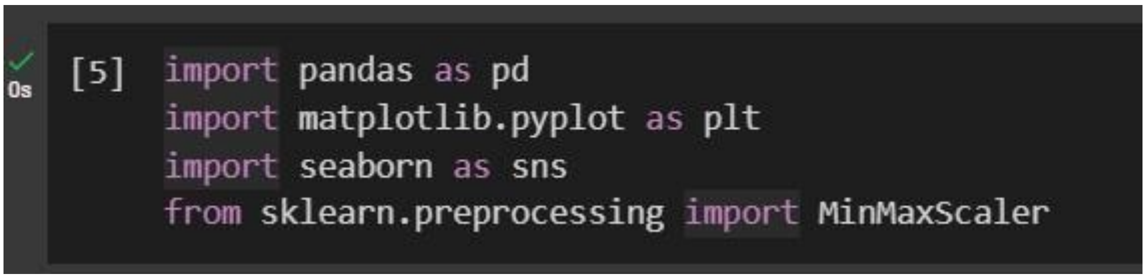
This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.Evaluating a machine learning or deep learning model is a crucial step in assessing its performance and determining how well it generalizes to new, unseen data. Model evaluation helps you understand whether the model's predictions are accurate and whether it's suitable for its intended task. Several evaluation metrics and techniques can be used to assess a model's performance.

**Accuracy**: Accuracy is a common metric for classification tasks. It measures the ratio of correctly predicted instances to the total number of instances in the test dataset.

**F1-Score**: The F1-score is the harmonic mean of precision and recall. It provides a balance between these two metrics and is particularly useful when dealing with imbalanced datasets.

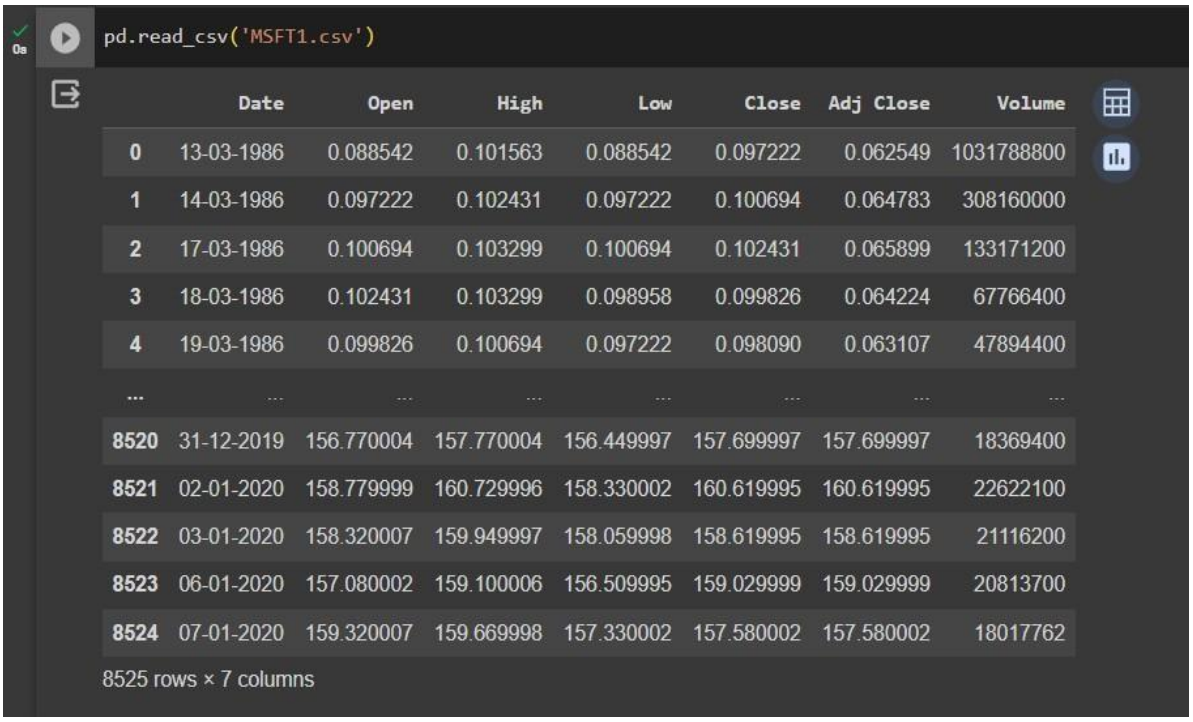
## TRANSFORMING INTO CODE:

### Import Libraries



```
[5] import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.preprocessing import MinMaxScaler
```

### Load the Dataset

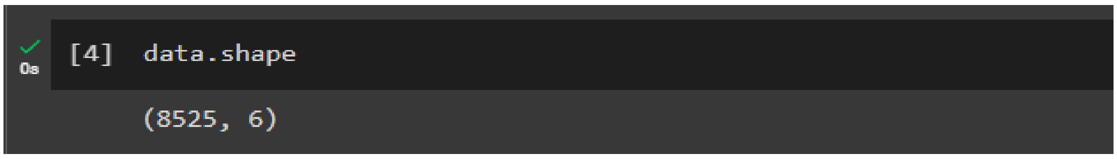


```
pd.read_csv('MSFT1.csv')
```

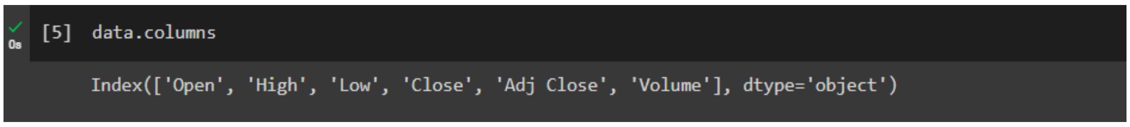
| | Date | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|---|
| 0 | 13-03-1986 | 0.088542 | 0.101563 | 0.088542 | 0.097222 | 0.062549 | 1031788800 |
| 1 | 14-03-1986 | 0.097222 | 0.102431 | 0.097222 | 0.100694 | 0.064783 | 308160000 |
| 2 | 17-03-1986 | 0.100694 | 0.103299 | 0.100694 | 0.102431 | 0.065899 | 133171200 |
| 3 | 18-03-1986 | 0.102431 | 0.103299 | 0.098958 | 0.099826 | 0.064224 | 67766400 |
| 4 | 19-03-1986 | 0.099826 | 0.100694 | 0.097222 | 0.098090 | 0.063107 | 47894400 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 8520 | 31-12-2019 | 156.770004 | 157.770004 | 156.449997 | 157.699997 | 157.699997 | 18369400 |
| 8521 | 02-01-2020 | 158.779999 | 160.729996 | 158.330002 | 160.619995 | 160.619995 | 22622100 |
| 8522 | 03-01-2020 | 158.320007 | 159.949997 | 158.059998 | 158.619995 | 158.619995 | 21116200 |
| 8523 | 06-01-2020 | 157.080002 | 159.100006 | 156.509995 | 159.029999 | 159.029999 | 20813700 |
| 8524 | 07-01-2020 | 159.320007 | 159.669998 | 157.330002 | 157.580002 | 157.580002 | 18017762 |

8525 rows × 7 columns

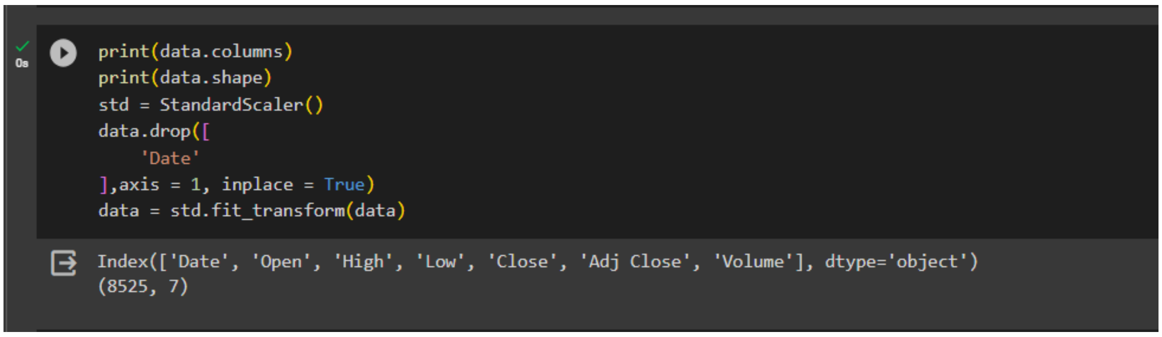
## Data Preparation



```
[4]  data.shape

     (8525, 6)
```

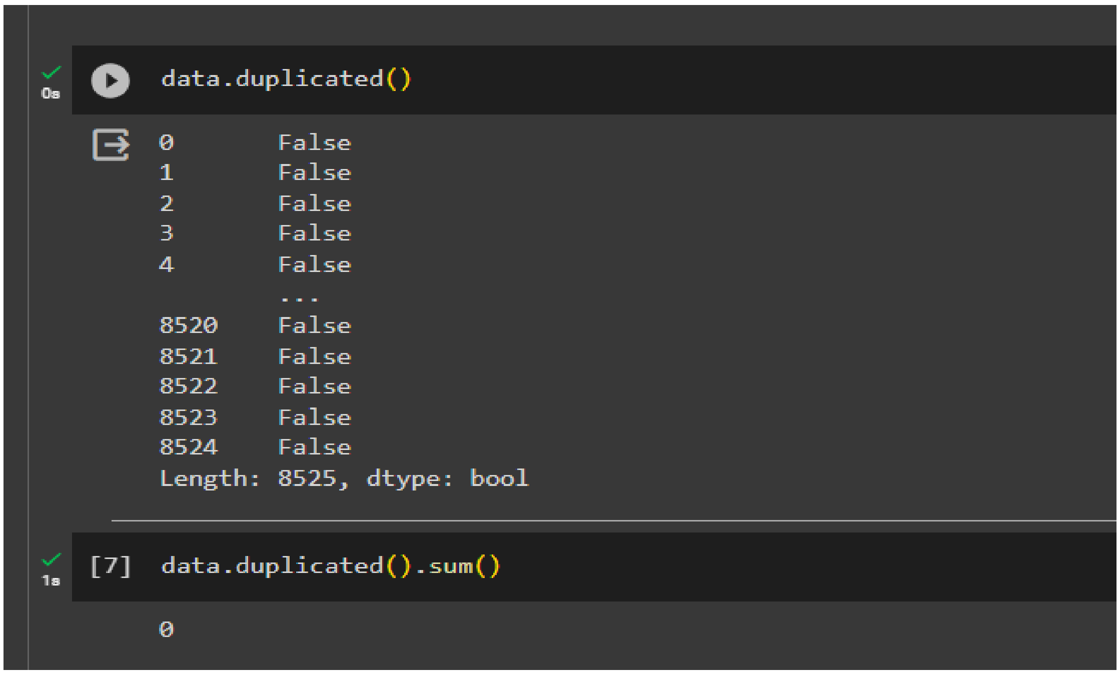


```
[5]  data.columns

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



```
print(data.columns)
print(data.shape)
std = StandardScaler()
data.drop([
    'Date'
],axis = 1, inplace = True)
data = std.fit_transform(data)
```

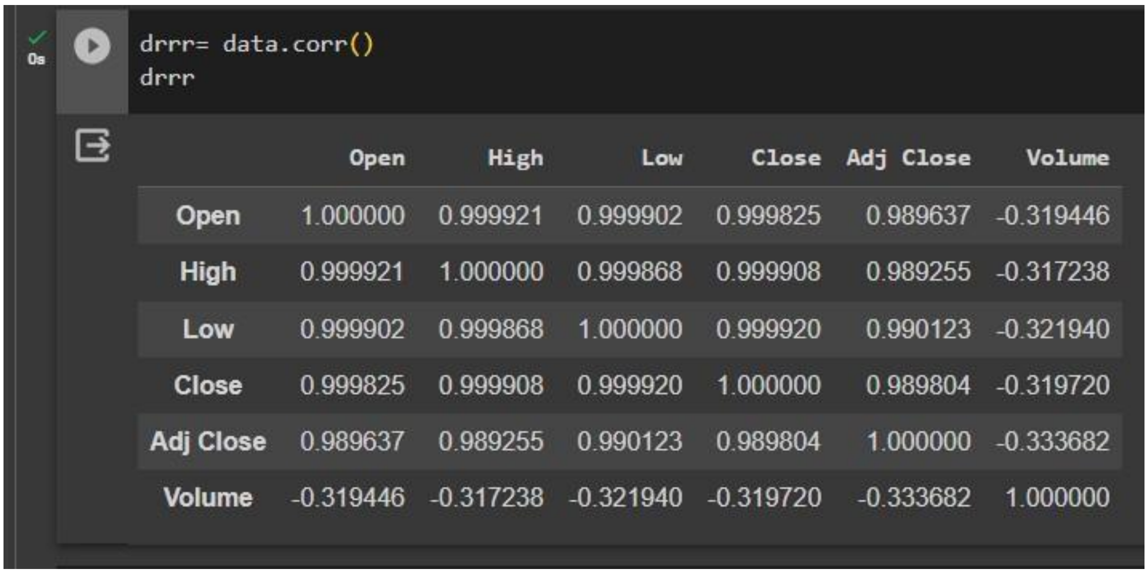
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], dtype='object')
(8525, 7)
```



```
data.duplicated()
```

```
0       False
1       False
2       False
3       False
4       False
        ...
8520    False
8521    False
8522    False
8523    False
8524    False
Length: 8525, dtype: bool
```

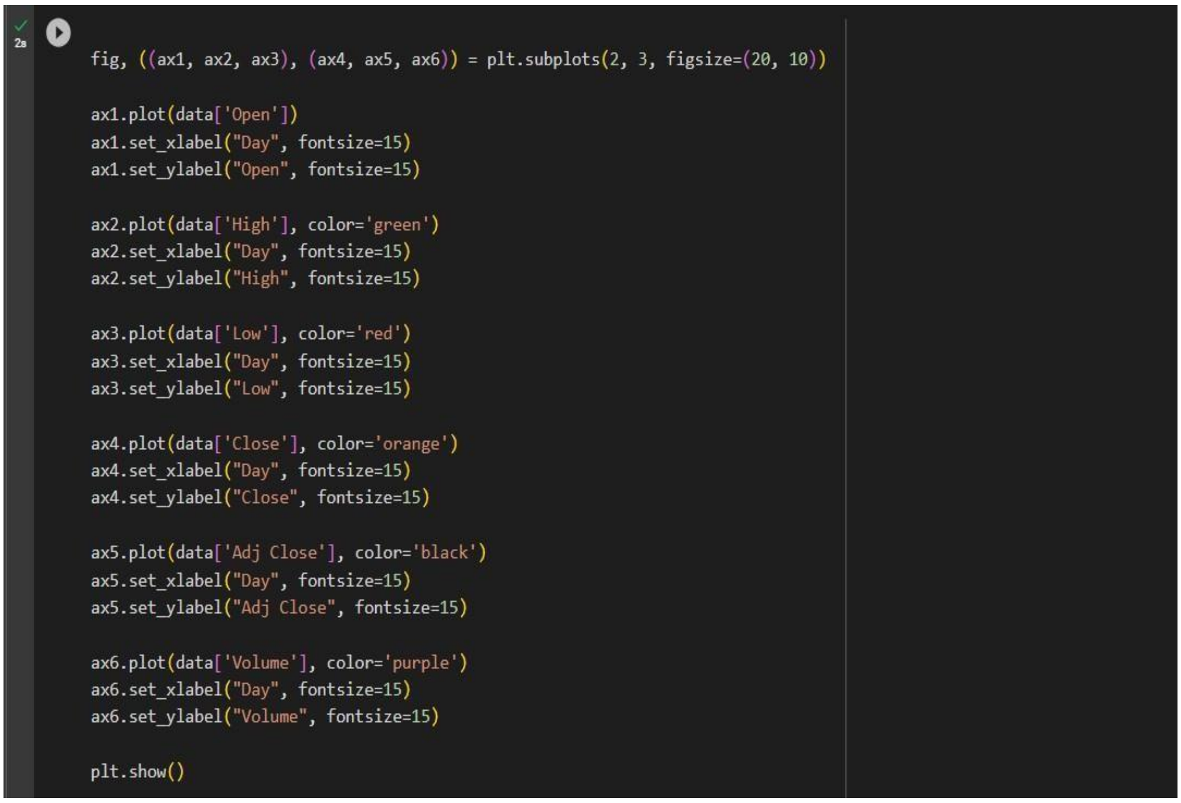
```
[7]  data.duplicated().sum()

     0
```



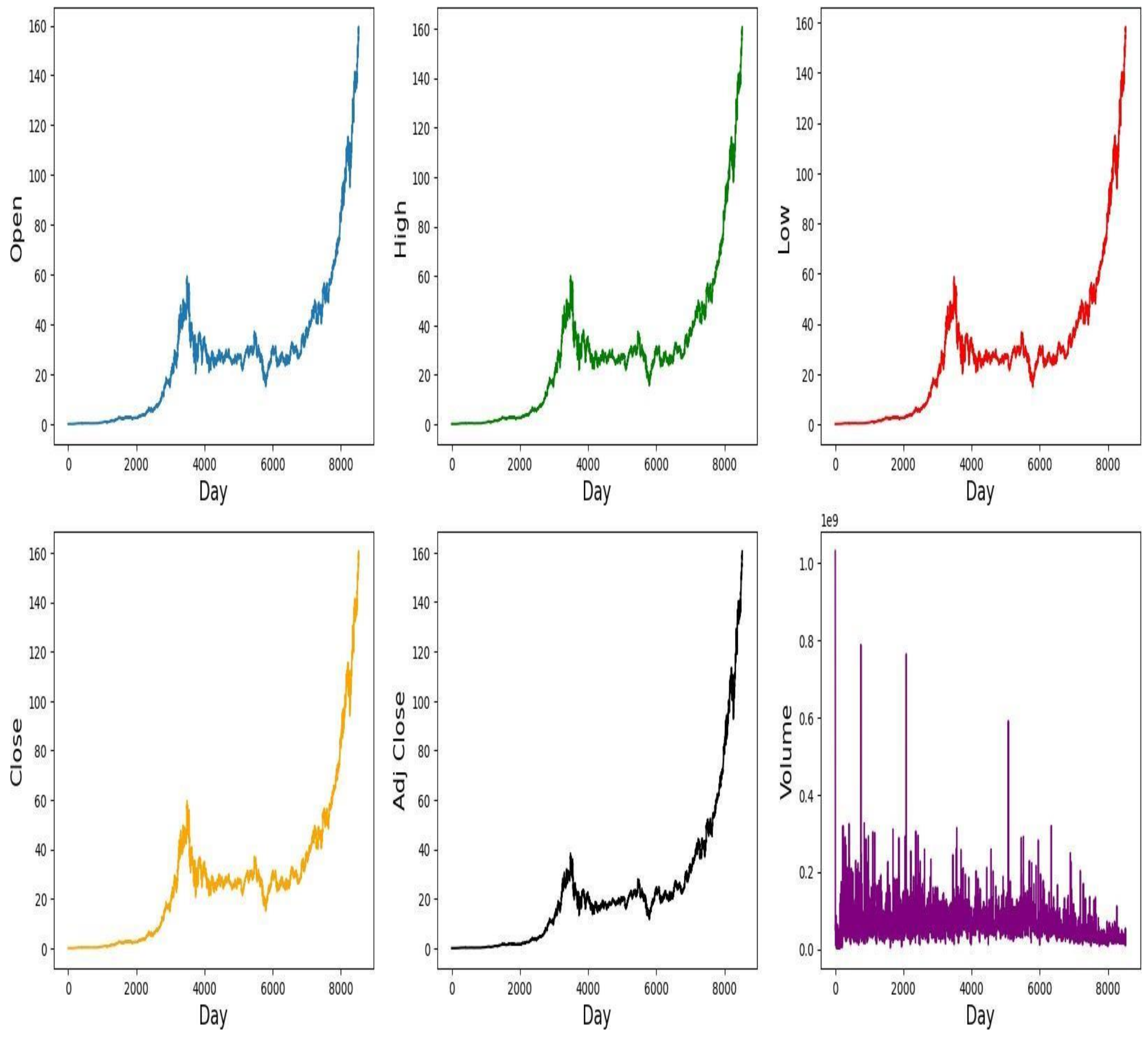
```
drrr= data.corr()
drrr
```

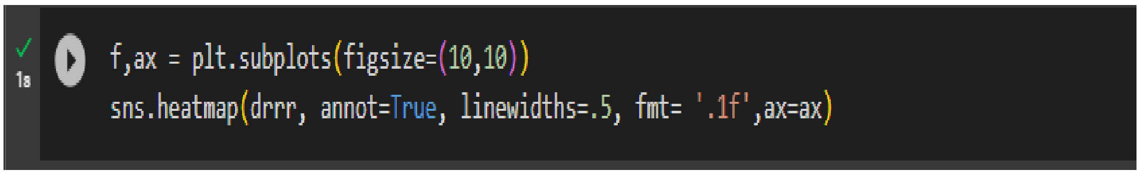
| | Open | High | Low | Close | Adj Close | Volume |
|---|---|---|---|---|---|---|
| Open | 1.000000 | 0.999921 | 0.999902 | 0.999825 | 0.989637 | -0.319446 |
| High | 0.999921 | 1.000000 | 0.999868 | 0.999908 | 0.989255 | -0.317238 |
| Low | 0.999902 | 0.999868 | 1.000000 | 0.999920 | 0.990123 | -0.321940 |
| Close | 0.999825 | 0.999908 | 0.999920 | 1.000000 | 0.989804 | -0.319720 |
| Adj Close | 0.989637 | 0.989255 | 0.990123 | 0.989804 | 1.000000 | -0.333682 |
| Volume | -0.319446 | -0.317238 | -0.321940 | -0.319720 | -0.333682 | 1.000000 |

## Data Visualization

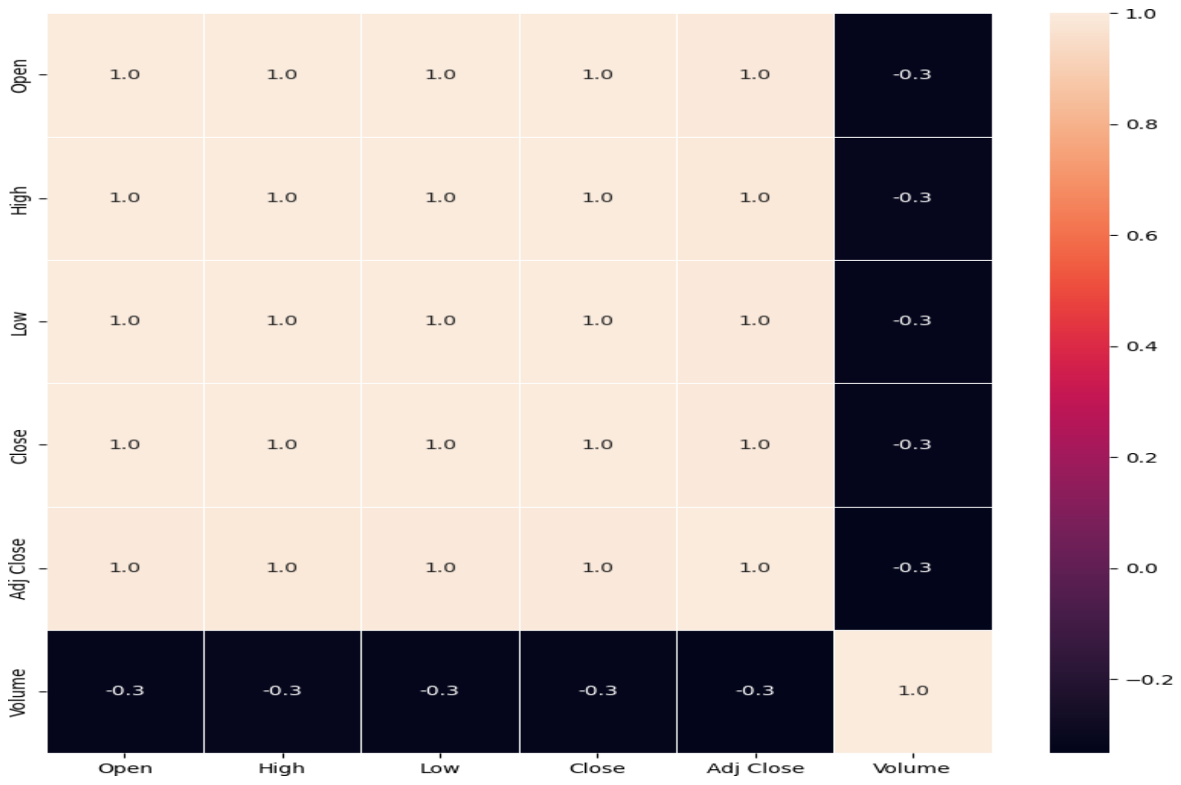


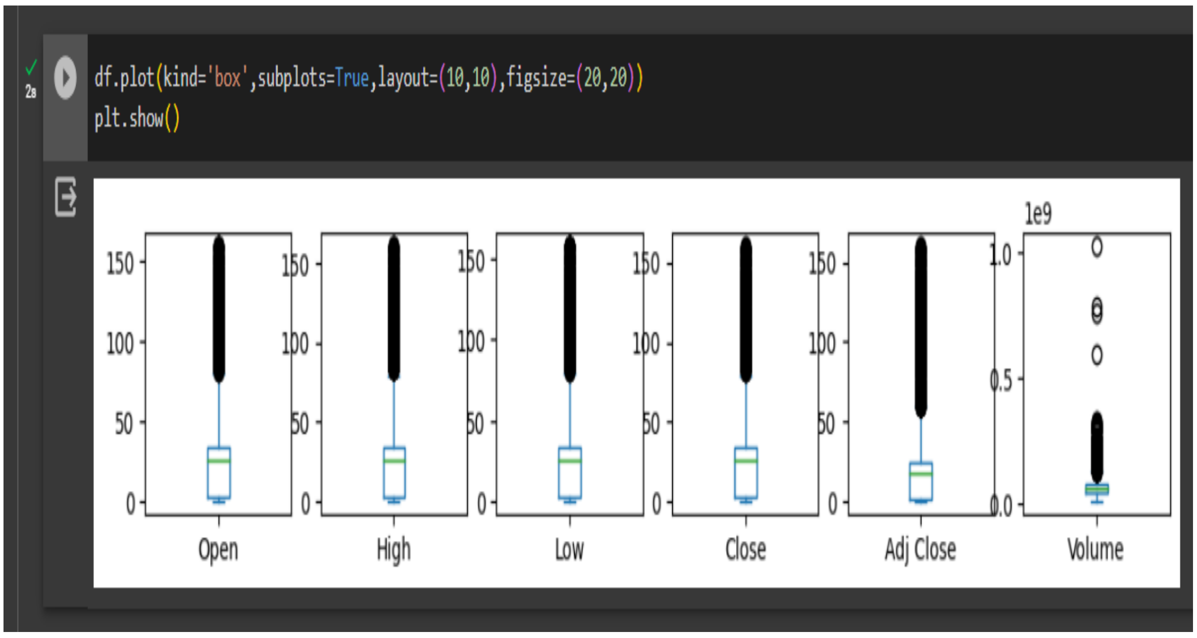
```
fig, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = plt.subplots(2, 3, figsize=(20, 10))

ax1.plot(data['Open'])
ax1.set_xlabel("Day", fontsize=15)
ax1.set_ylabel("Open", fontsize=15)

ax2.plot(data['High'], color='green')
ax2.set_xlabel("Day", fontsize=15)
ax2.set_ylabel("High", fontsize=15)

ax3.plot(data['Low'], color='red')
ax3.set_xlabel("Day", fontsize=15)
ax3.set_ylabel("Low", fontsize=15)

ax4.plot(data['Close'], color='orange')
ax4.set_xlabel("Day", fontsize=15)
ax4.set_ylabel("Close", fontsize=15)

ax5.plot(data['Adj Close'], color='black')
ax5.set_xlabel("Day", fontsize=15)
ax5.set_ylabel("Adj Close", fontsize=15)

ax6.plot(data['Volume'], color='purple')
ax6.set_xlabel("Day", fontsize=15)
ax6.set_ylabel("Volume", fontsize=15)

plt.show()
```



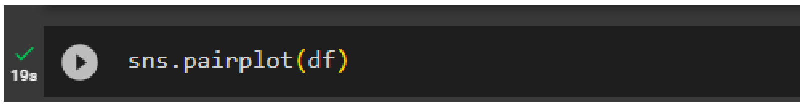


```
f,ax = plt.subplots(figsize=(10,10))
sns.heatmap(drrr, annot=True, linewidths=.5, fmt= '.1f',ax=ax)
```

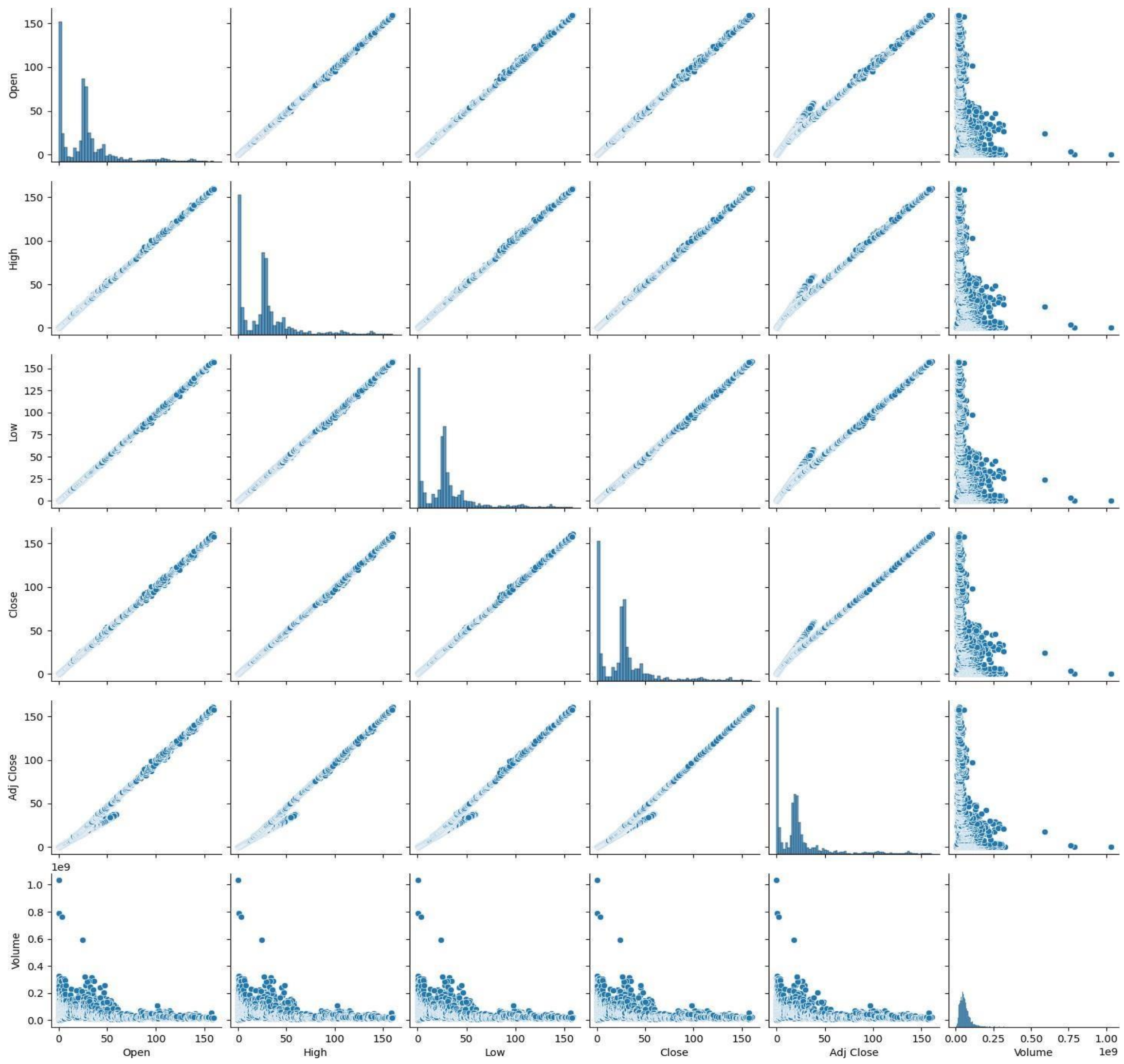


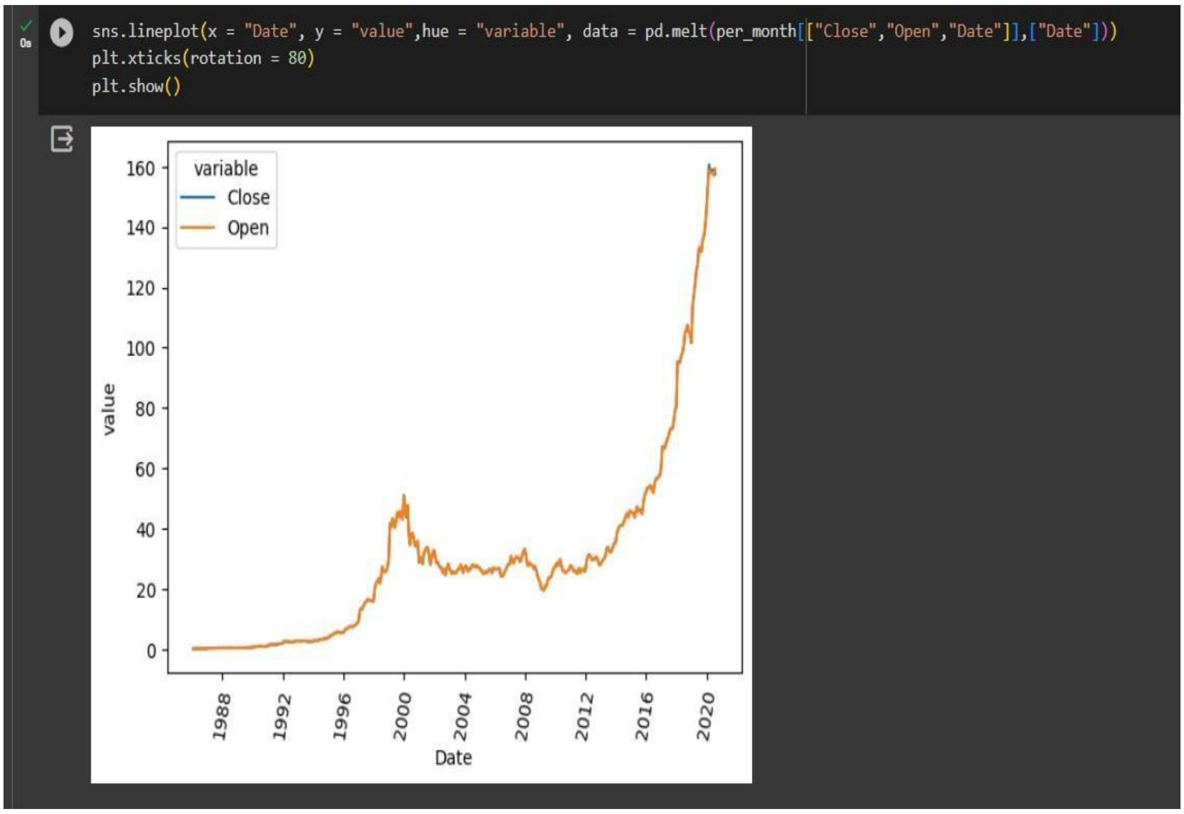


```
df.plot(kind='box',subplots=True,layout=(10,10),figsize=(20,20))
plt.show()
```



```
sns.pairplot(df)
```





```
sns.lineplot(x = "Date", y = "value",hue = "variable", data = pd.melt(per_month[["Close","Open","Date"]],["Date"]))
plt.xticks(rotation = 80)
plt.show()
```

## PROCEDURE

### Feature selection:

#### 1. Identify the target variable:

This is the variable that you want to predict, such as house price.

#### 2. Explore the data.

This will help you to understand the relationships between the differentfeatures and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

#### 3. Remove redundant features.

If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.

#### 4. Remove irrelevant features.

If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

### Model Training

- ❖ Choose a machine learning algorithm.
- ❖ There are a number of different machine learning algorithms that can be used for stock price prediction, such as linear regression, LSTM, KNN, ridge regression, lasso regression, decision trees, and random forests are Covered above.

## Model evaluation:

- ❖ Model evaluation is the process of assessing the performance of a machine learning model on unseen data. This is important to ensure that the model will generalize well to new data.
- ❖ There are a number of different metrics that can be used to evaluate the performance of a house price prediction model. Some of the most common metrics include:

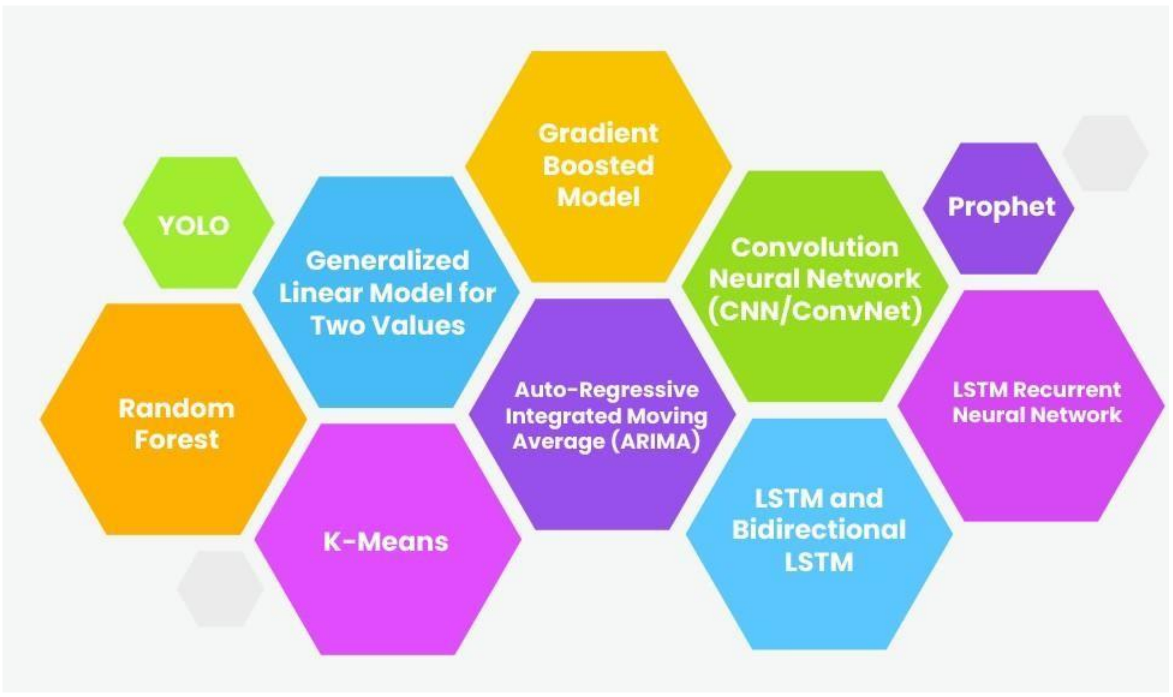
- **Mean squared error (MSE):** This metric measures the average squared difference between the predicted and actual house prices.

- **Root mean squared error (RMSE):** This metric is the square root of the MSE.

- **Mean absolute error (MAE):** This metric measures the average absolute difference between the predicted and actual house prices.

- **R-squared:** This metric measures how well the model explains the variation in the actual house prices.

## PREDICTIVE ANALYSIS TECHINIQUES


YOLO
Gradient Boosted Model
Prophet
Generalized Linear Model for Two Values
Convolution Neural Network (CNN/ConvNet)
Random Forest
Auto-Regressive Integrated Moving Average (ARIMA)
LSTM Recurrent Neural Network
K-Means
LSTM and Bidirectional LSTM

### Some of the techniques are:

Overall, predictive analytics algorithms can be separated into twogroups: machine learning and deep learning.

- **Machine learning** involves structural data that we see in atable. Algorithms for this comprise both linear and nonlinear varieties. Linear algorithms train more quickly, while nonlinear are better optimized for the problems theyare likely to face (which are often nonlinear).
- **Deep learning** is a subset of machine learning that is morepopular to deal with audio, video, text, and images.

With machine learning predictive modelling, there are several different algorithms that can be applied. Below are some of the mostcommon algorithms that are being used to power the predictive analytics models described above.

## 1. LSTM

- LSTMs are a type of Recurrent Neural Network (RNN) that can learn and memorize long-term dependencies.
- LSTM is designed to handle sequential data with long-term dependencies.
- It includes memory cells that can store and retrieve informationover extended sequences.
- LSTMs are effective at capturing patterns and relationships insequential data.
- They can be used for various tasks, including text generation,sentiment.
- Random Forest, XGBoost, and LSTM (Long Short-Term Memory) are three distinct machine learning models, each withits own characteristics and use cases.
- They can be applied to a wide range of problems, includingclassification, regression, and time series forecasting.

## 2. Random Forest:

- Random Forest is an ensemble of decision trees.
- It combines the predictions of multiple decision trees to producea more accurate and stable prediction.
- It is capable of handling both categorical and numerical features.
- Random Forest provides feature importance scores, which canhelp identify the most important features in the dataset.
- It is less prone to overfitting compared to individual decision trees.

## 3. XGBoost (Extreme Gradient Boosting):

- XGBoost is a highly efficient and scalable implementation of gradient boosting machines.
- It is known for its superior performance on a wide range ofmachine learning tasks.
- XGBoost allows for custom loss functions and optimization objectives.
- It can handle missing data and is robust to outliers.
- It offers feature selection through importance scores.

## 4. Generalized Linear Model

- The generalized linear model is a complex extension of the general linear model. It takes the latter model's comparison ofthe effects of multiple variables on continuous variables. Afterthat, it draws from various distributions to find the "best fit" model.
- The most important advantage of this predictive model is that ittrains very quickly. Also, it helps to deal with the categorical predictors as it is simple to interpret. A generalized linear modelhelps understand how the predictors will affect future outcomesand resist overfitting.

## 5. Prophet

- The Prophet algorithm is generally used in forecast models and time series models. This predictive analytics algorithm was initially developed by Facebook and is used internally by the company for forecasting.
- The Prophet algorithm is excellent for capacity planning by automatically allocating the resources and setting appropriate sales goals.
- Manual forecasting of data requires hours of labour work with highly professional analysts to draw out accurate outputs. With inconsistent performance levels and inflexibility of other forecasting algorithms, the prophet algorithm is a valuable alternative.

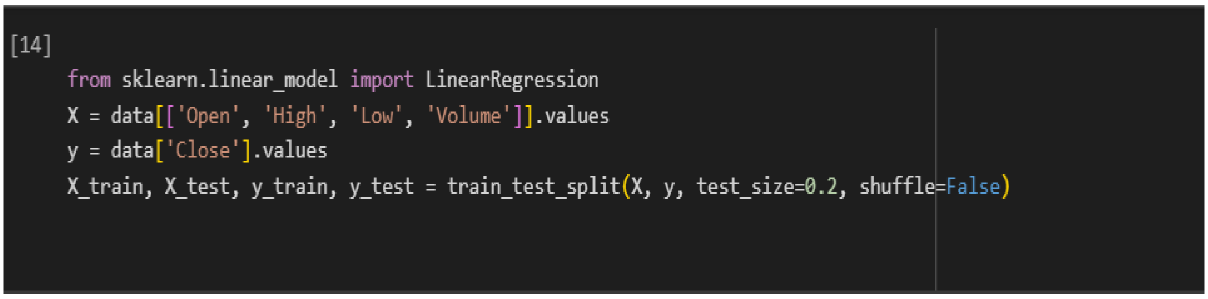
## 6. Convolution Neural Network (CNN/ConvNet)

- Convolution neural networks(CNN) is artificial neural network that performs feature detection in image data.
- They are based on the convolution operation, transforming the input image into a matrix where rows and columns correspond todifferent image planes and differentiate one object.
- The architecture of the CNN model is inspired by the visual cortex of the human brain. As a result, it is quite similar to the pattern of neurons connected in the human brain. Individual neurons of the model respond to stimuli only to specific regions of the visual field known as the Receptive Field.

## TECHNIQUES

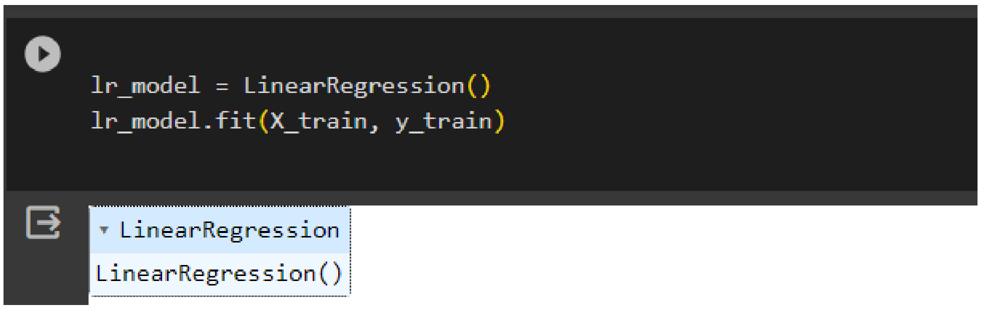
## 1. Linear Regression

### Data Preparation



```
[14]
    from sklearn.linear_model import LinearRegression
    X = data[['Open', 'High', 'Low', 'Volume']].values
    y = data['Close'].values
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

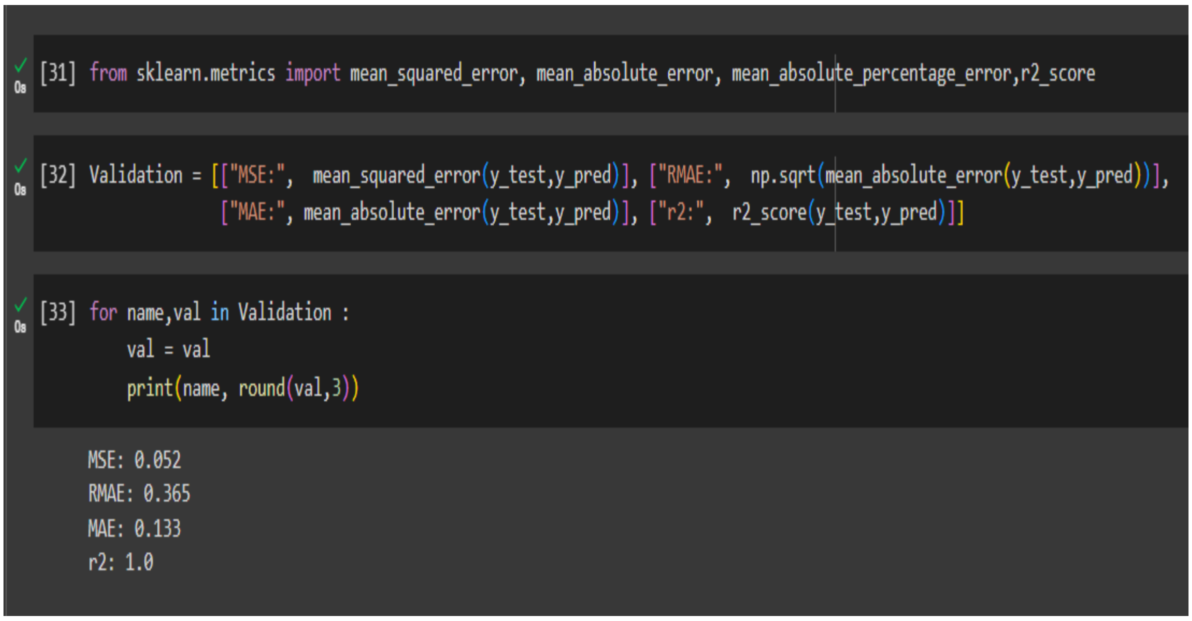
### Model training



```
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

### Model Evaluation



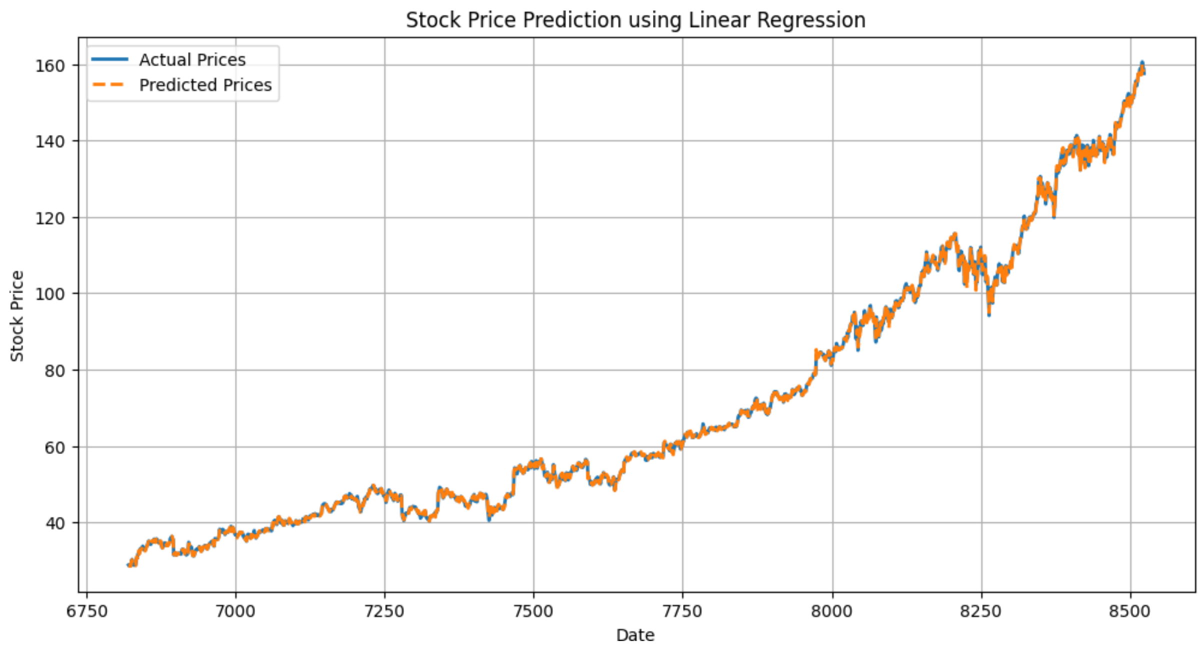
```
[31] from sklearn.metrics import mean_squared_error, mean_absolute_error, mean_absolute_percentage_error,r2_score
```

```
[32] Validation = [["MSE:",  mean_squared_error(y_test,y_pred)], ["RMAE:",  np.sqrt(mean_absolute_error(y_test,y_pred))],
                ["MAE:", mean_absolute_error(y_test,y_pred)], ["r2:",  r2_score(y_test,y_pred)]]
```

```
[33] for name,val in Validation :
         val = val
         print(name, round(val,3))
```

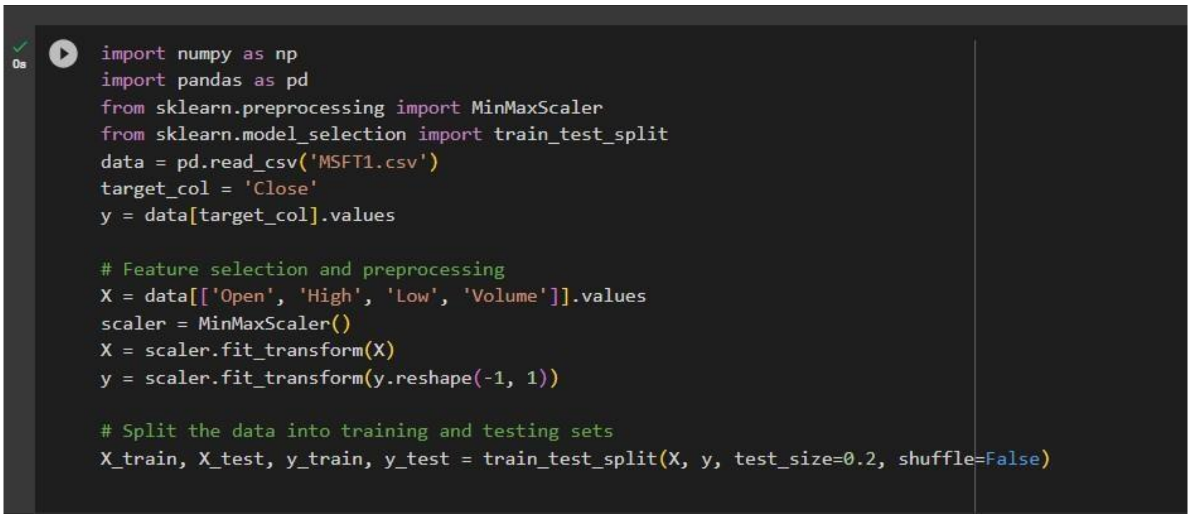
```
MSE: 0.052
RMAE: 0.365
MAE: 0.133
r2: 1.0
```

## Visualization of Linear regression



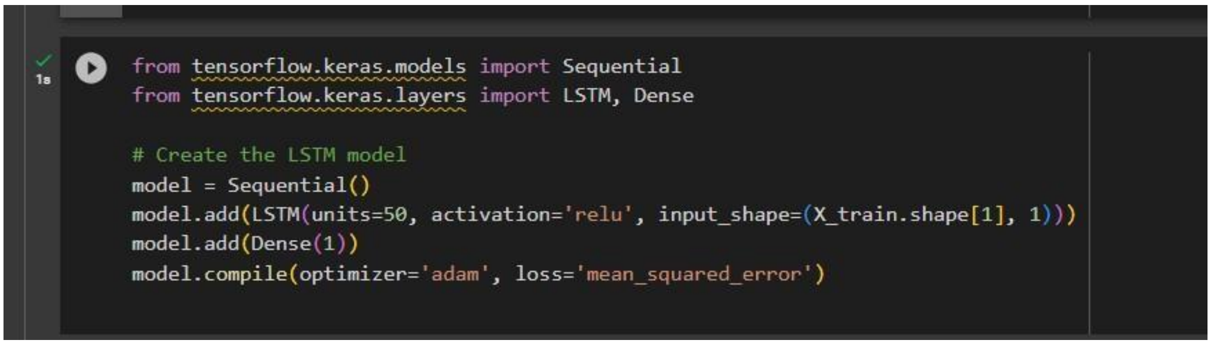
## 2.LSTM

## Data preparation



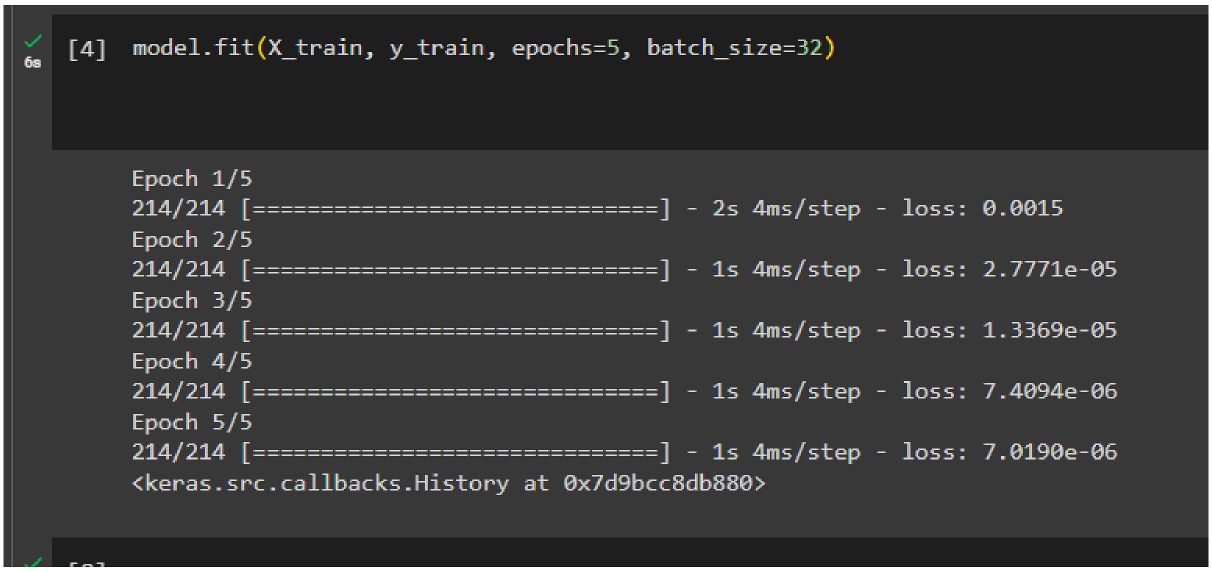
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
data = pd.read_csv('MSFT1.csv')
target_col = 'Close'
y = data[target_col].values

# Feature selection and preprocessing
X = data[['Open', 'High', 'Low', 'Volume']].values
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
y = scaler.fit_transform(y.reshape(-1, 1))

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

## LSTM Model Creation



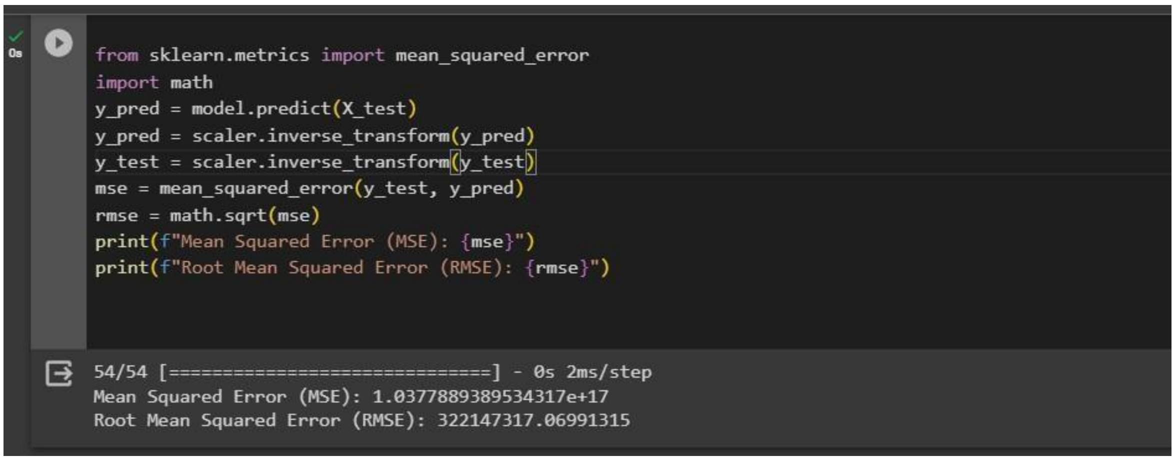
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense

# Create the LSTM model
model = Sequential()
model.add(LSTM(units=50, activation='relu', input_shape=(X_train.shape[1], 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

## Model training



```
[4] model.fit(X_train, y_train, epochs=5, batch_size=32)

Epoch 1/5
214/214 [==============================] - 2s 4ms/step - loss: 0.0015
Epoch 2/5
214/214 [==============================] - 1s 4ms/step - loss: 2.7771e-05
Epoch 3/5
214/214 [==============================] - 1s 4ms/step - loss: 1.3369e-05
Epoch 4/5
214/214 [==============================] - 1s 4ms/step - loss: 7.4094e-06
Epoch 5/5
214/214 [==============================] - 1s 4ms/step - loss: 7.0190e-06
<keras.src.callbacks.History at 0x7d9bcc8db880>
```

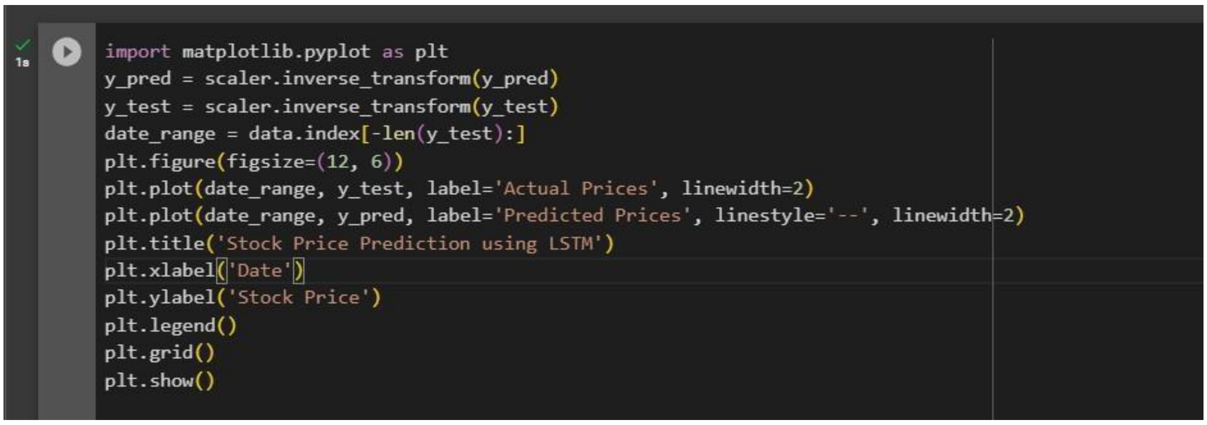
## Model Evaluation



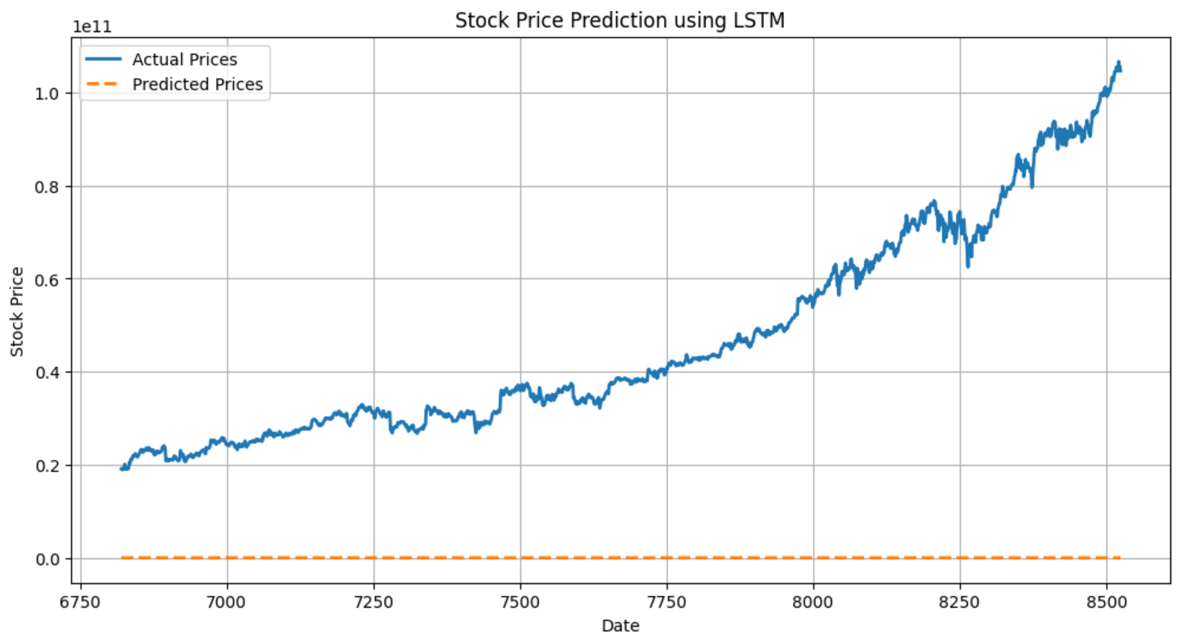
```
from sklearn.metrics import mean_squared_error
import math
y_pred = model.predict(X_test)
y_pred = scaler.inverse_transform(y_pred)
y_test = scaler.inverse_transform(y_test)
mse = mean_squared_error(y_test, y_pred)
rmse = math.sqrt(mse)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
54/54 [==============================] - 0s 2ms/step
Mean Squared Error (MSE): 1.0377889389534317e+17
Root Mean Squared Error (RMSE): 322147317.06991315
```

## Visualization of LSTM Model

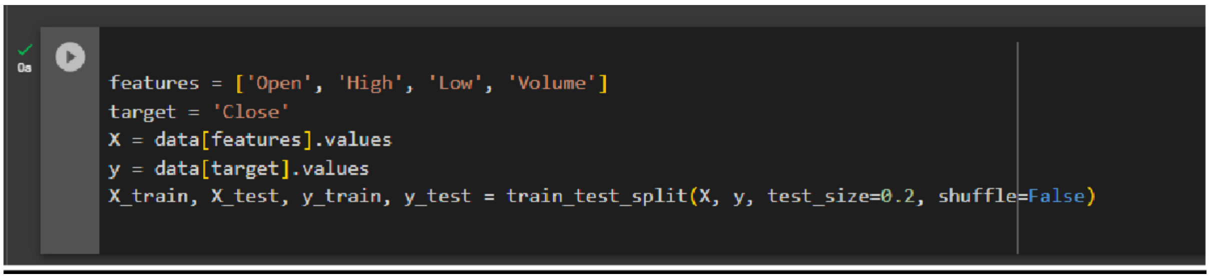


```
import matplotlib.pyplot as plt
y_pred = scaler.inverse_transform(y_pred)
y_test = scaler.inverse_transform(y_test)
date_range = data.index[-len(y_test):]
plt.figure(figsize=(12, 6))
plt.plot(date_range, y_test, label='Actual Prices', linewidth=2)
plt.plot(date_range, y_pred, label='Predicted Prices', linestyle='--', linewidth=2)
plt.title('Stock Price Prediction using LSTM')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.grid()
plt.show()
```



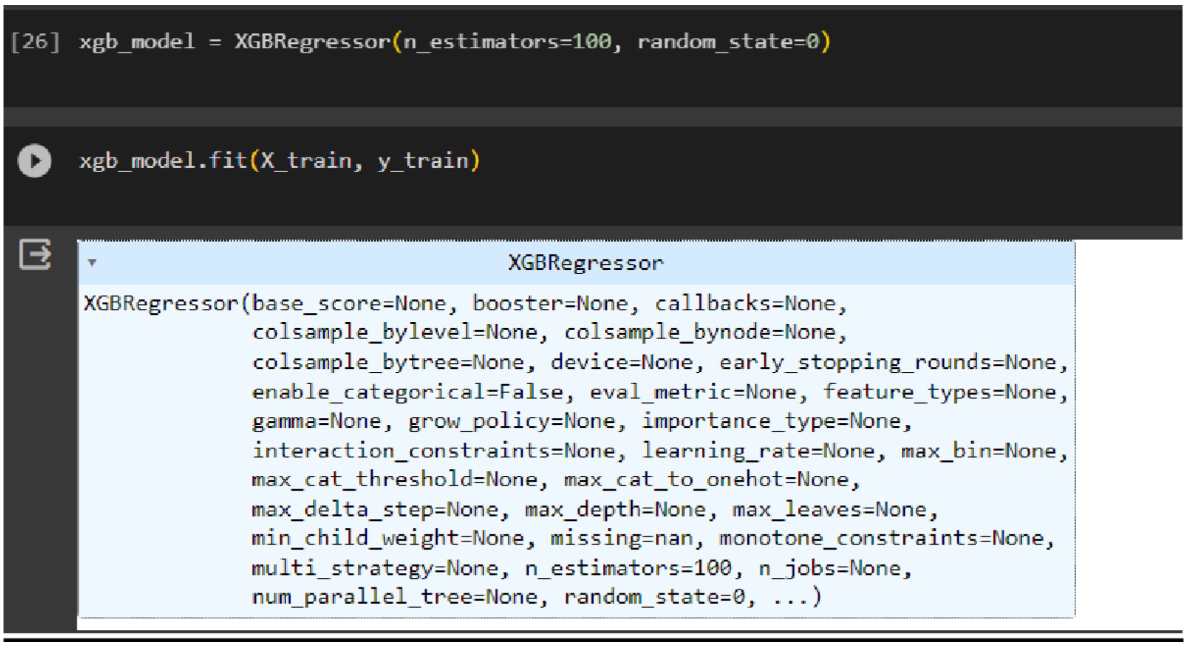
## 3.XGBoost

### Data Preparation



```
features = ['Open', 'High', 'Low', 'Volume']
target = 'Close'
X = data[features].values
y = data[target].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

### XGBoost Model Building and Training

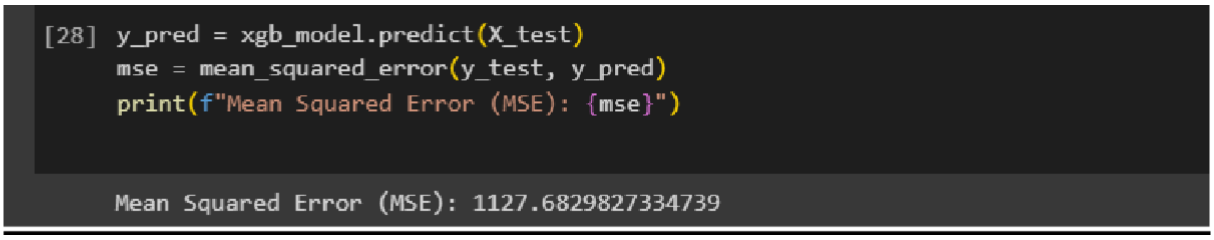


```
[26] xgb_model = XGBRegressor(n_estimators=100, random_state=0)
```

```
xgb_model.fit(X_train, y_train)
```

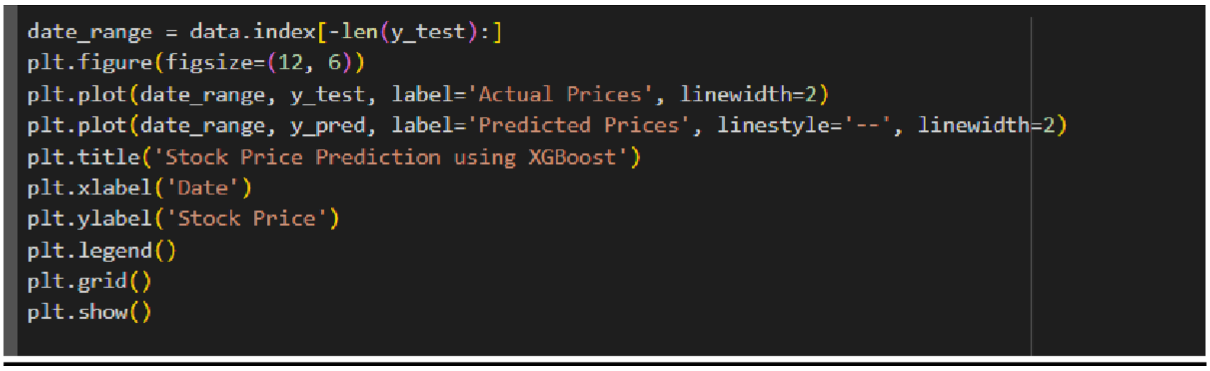
```
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=0, ...)
```

## Model Evaluation

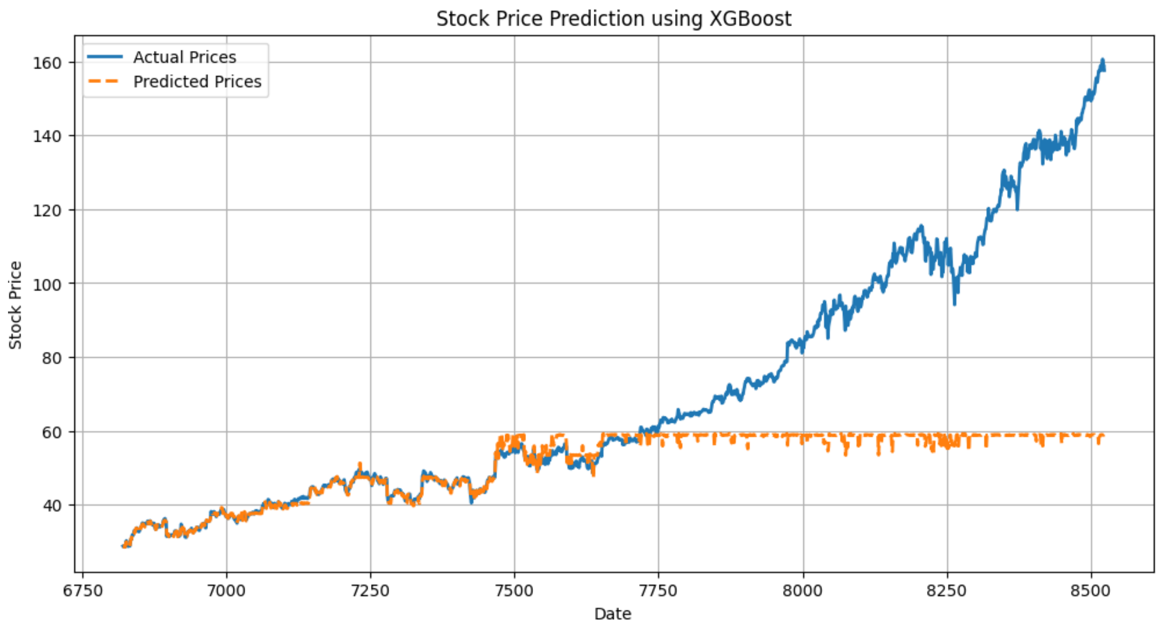


```
[28] y_pred = xgb_model.predict(X_test)
     mse = mean_squared_error(y_test, y_pred)
     print(f"Mean Squared Error (MSE): {mse}")

     Mean Squared Error (MSE): 1127.6829827334739
```

### Visualization of XGBoost

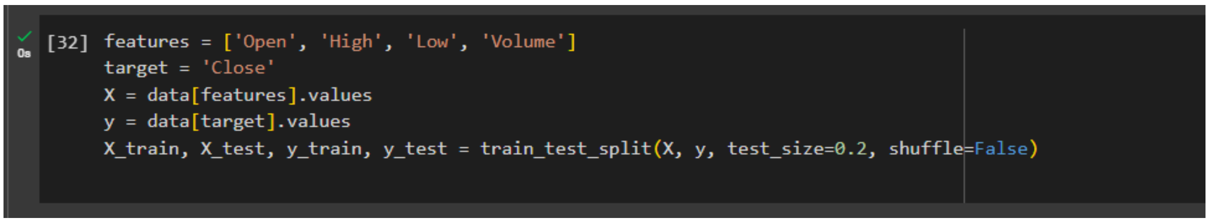


```
date_range = data.index[-len(y_test):]
plt.figure(figsize=(12, 6))
plt.plot(date_range, y_test, label='Actual Prices', linewidth=2)
plt.plot(date_range, y_pred, label='Predicted Prices', linestyle='--', linewidth=2)
plt.title('Stock Price Prediction using XGBoost')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.grid()
plt.show()
```



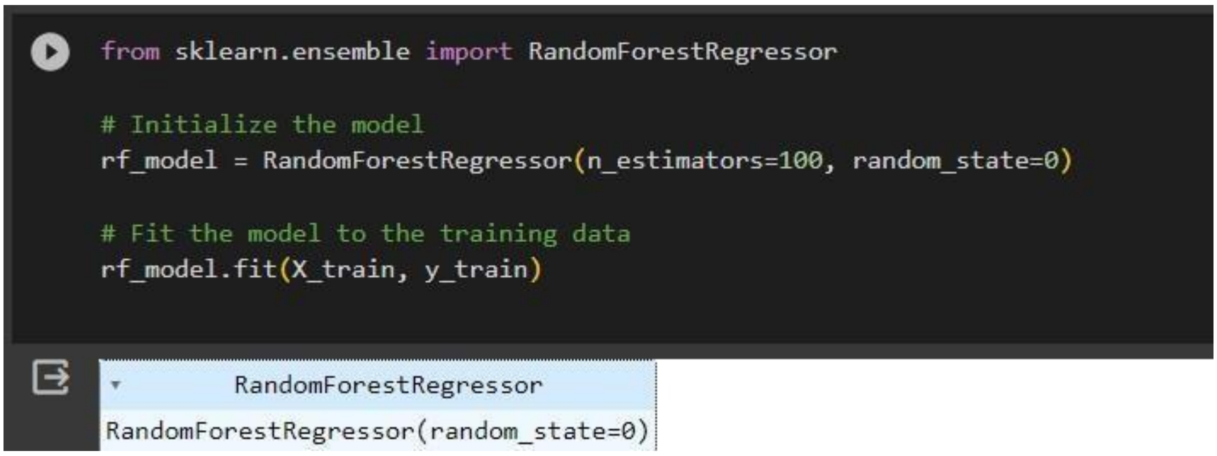
## 4. Random Forest

### Data Preparation



```
[32] features = ['Open', 'High', 'Low', 'Volume']
     target = 'Close'
     X = data[features].values
     y = data[target].values
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

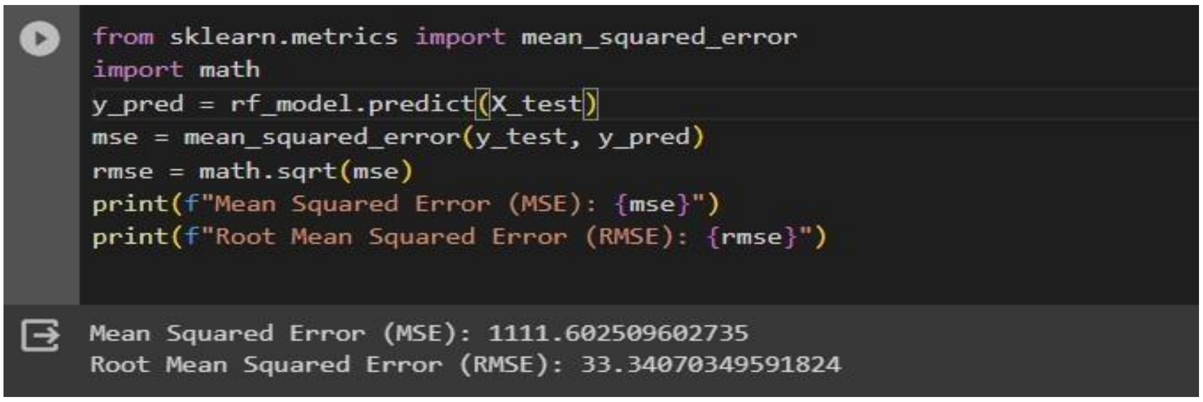
### Model Training



```
from sklearn.ensemble import RandomForestRegressor

# Initialize the model
rf_model = RandomForestRegressor(n_estimators=100, random_state=0)

# Fit the model to the training data
rf_model.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=0)
```

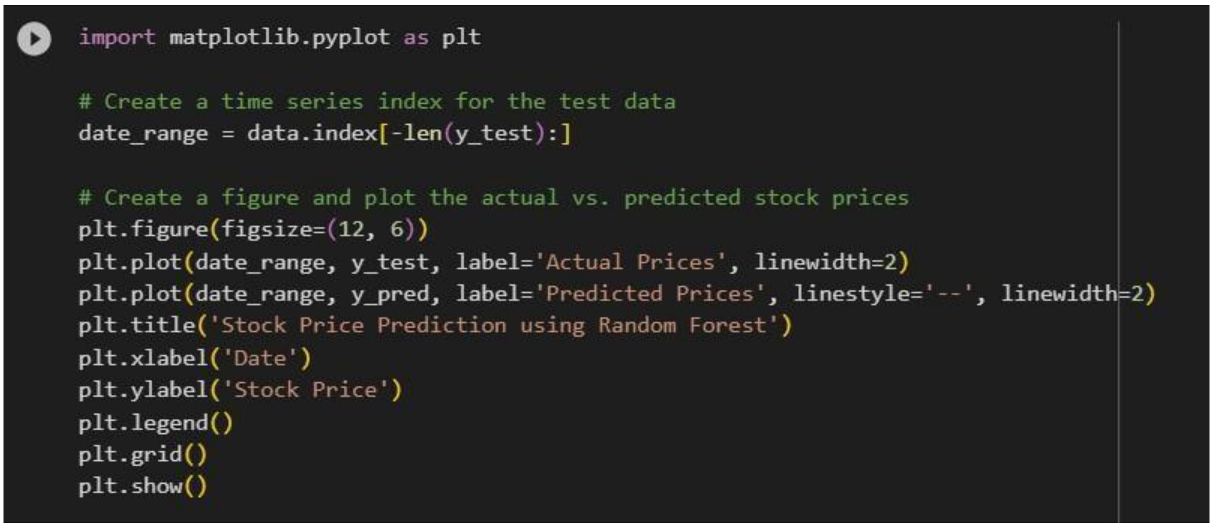
### Model Evaluation



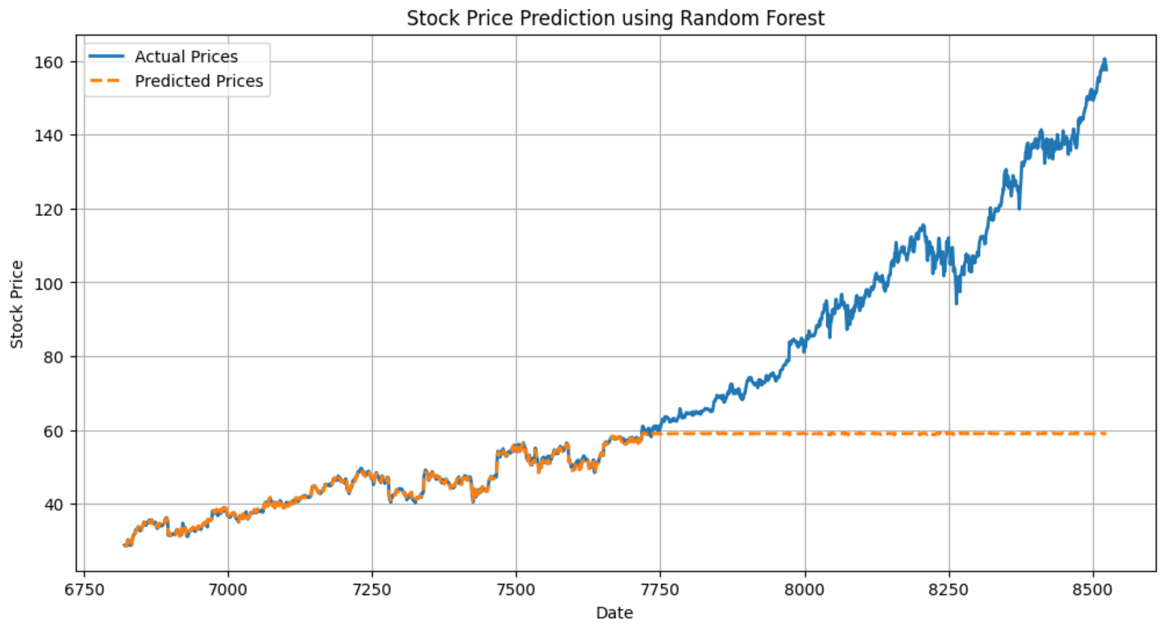
```
from sklearn.metrics import mean_squared_error
import math
y_pred = rf_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = math.sqrt(mse)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Mean Squared Error (MSE): 1111.602509602735
Root Mean Squared Error (RMSE): 33.34070349591824
```

## Visualization of Random Forest

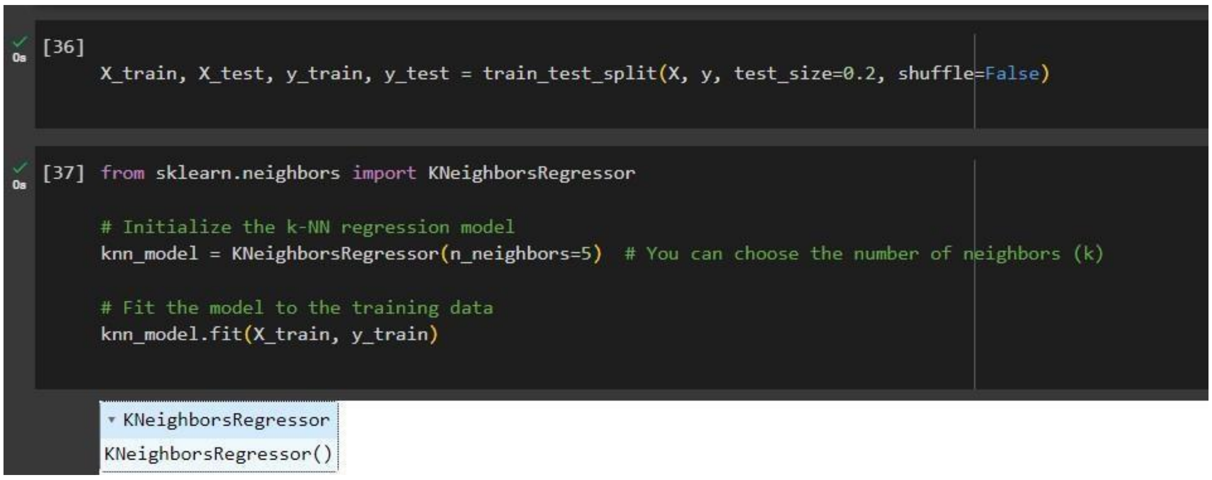


```
import matplotlib.pyplot as plt

# Create a time series index for the test data
date_range = data.index[-len(y_test):]

# Create a figure and plot the actual vs. predicted stock prices
plt.figure(figsize=(12, 6))
plt.plot(date_range, y_test, label='Actual Prices', linewidth=2)
plt.plot(date_range, y_pred, label='Predicted Prices', linestyle='--', linewidth=2)
plt.title('Stock Price Prediction using Random Forest')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.grid()
plt.show()
```



## 5. KNN

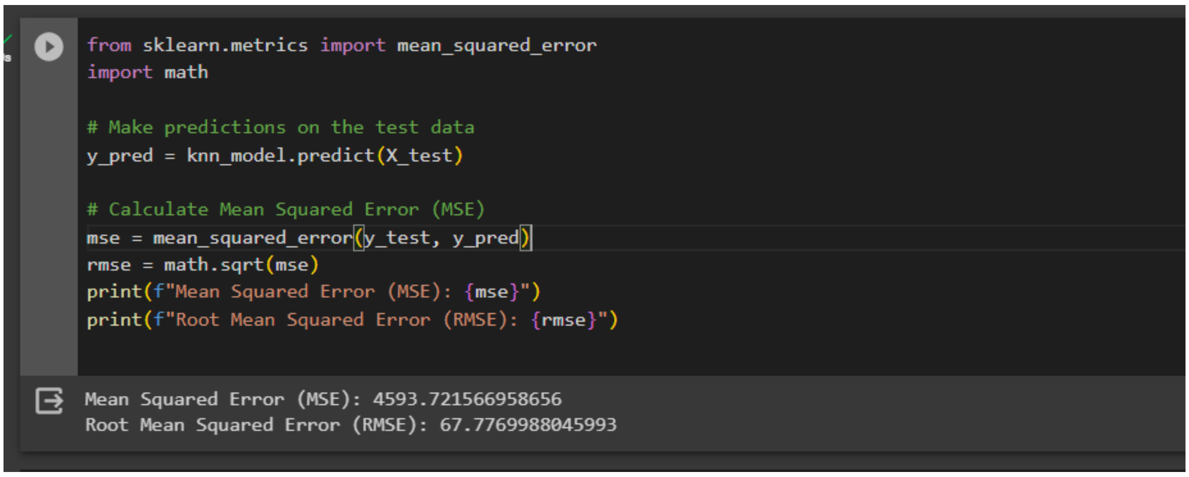
### Model Training



```
[36]
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

[37] from sklearn.neighbors import KNeighborsRegressor

     # Initialize the k-NN regression model
     knn_model = KNeighborsRegressor(n_neighbors=5)  # You can choose the number of neighbors (k)

     # Fit the model to the training data
     knn_model.fit(X_train, y_train)
```

```
▾ KNeighborsRegressor
KNeighborsRegressor()
```

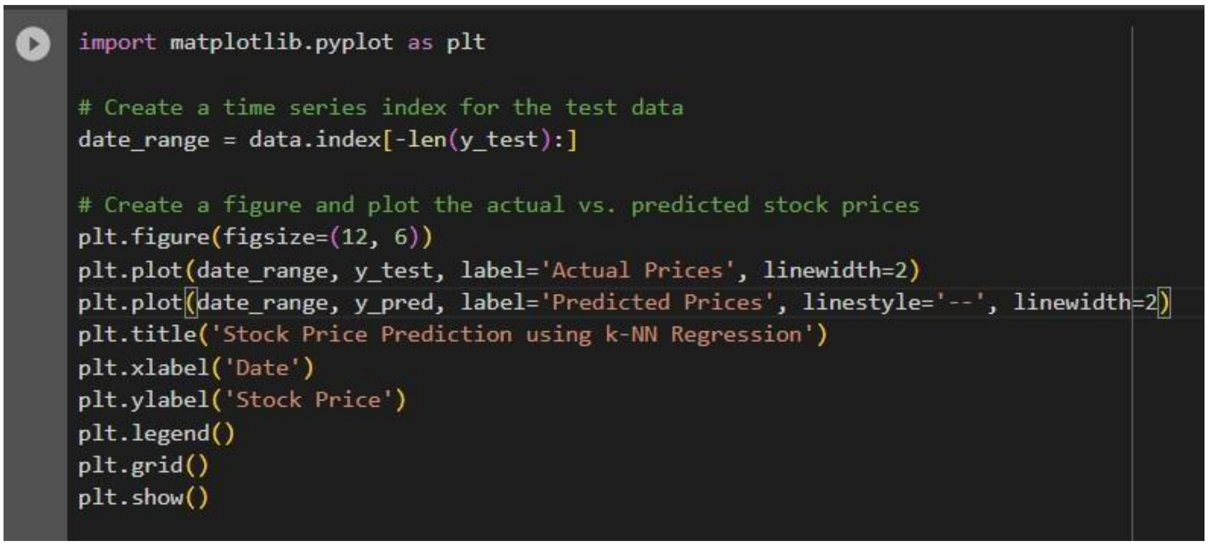
### Model Evaluation



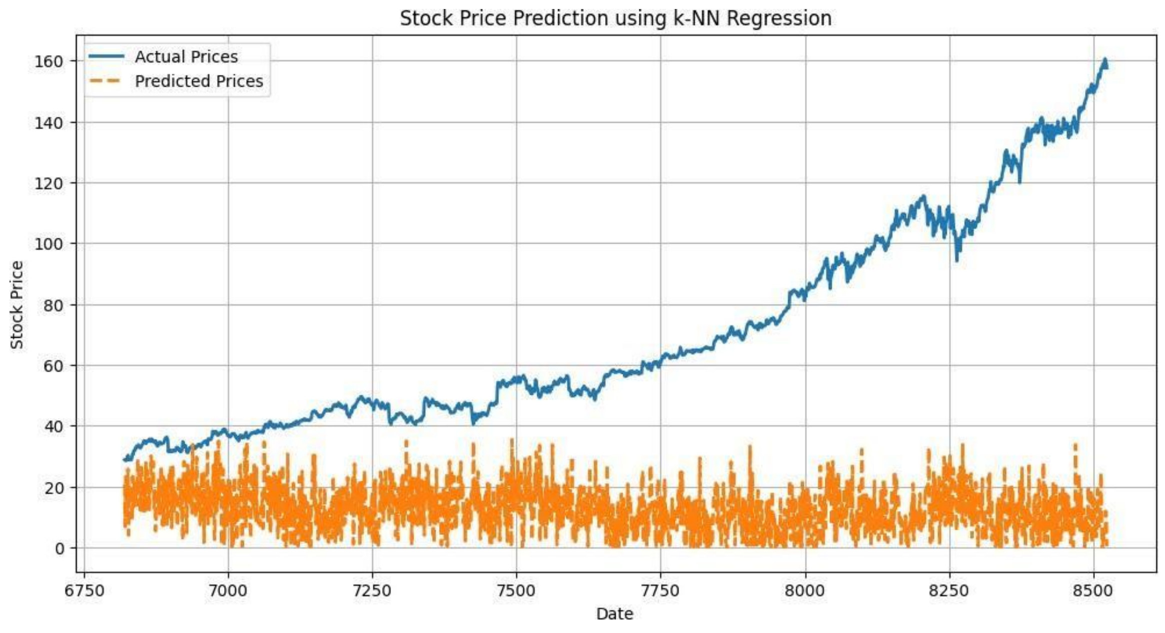
```
from sklearn.metrics import mean_squared_error
import math

# Make predictions on the test data
y_pred = knn_model.predict(X_test)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
rmse = math.sqrt(mse)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

```
Mean Squared Error (MSE): 4593.721566958656
Root Mean Squared Error (RMSE): 67.7769988045993
```

## Visualization of KNN

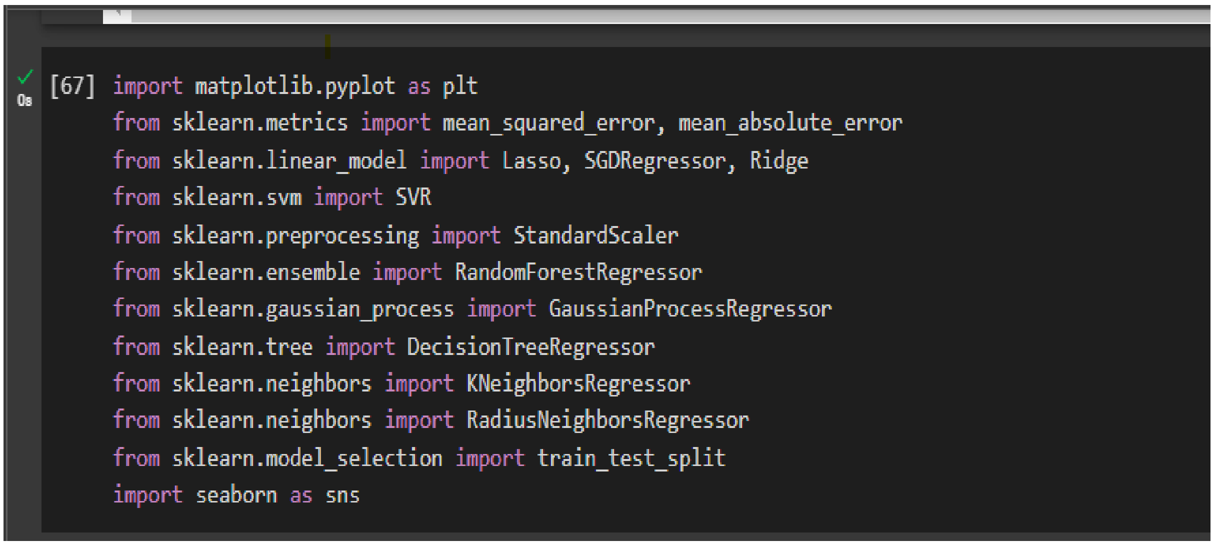


```
import matplotlib.pyplot as plt

# Create a time series index for the test data
date_range = data.index[-len(y_test):]

# Create a figure and plot the actual vs. predicted stock prices
plt.figure(figsize=(12, 6))
plt.plot(date_range, y_test, label='Actual Prices', linewidth=2)
plt.plot(date_range, y_pred, label='Predicted Prices', linestyle='--', linewidth=2)
plt.title('Stock Price Prediction using k-NN Regression')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.legend()
plt.grid()
plt.show()
```



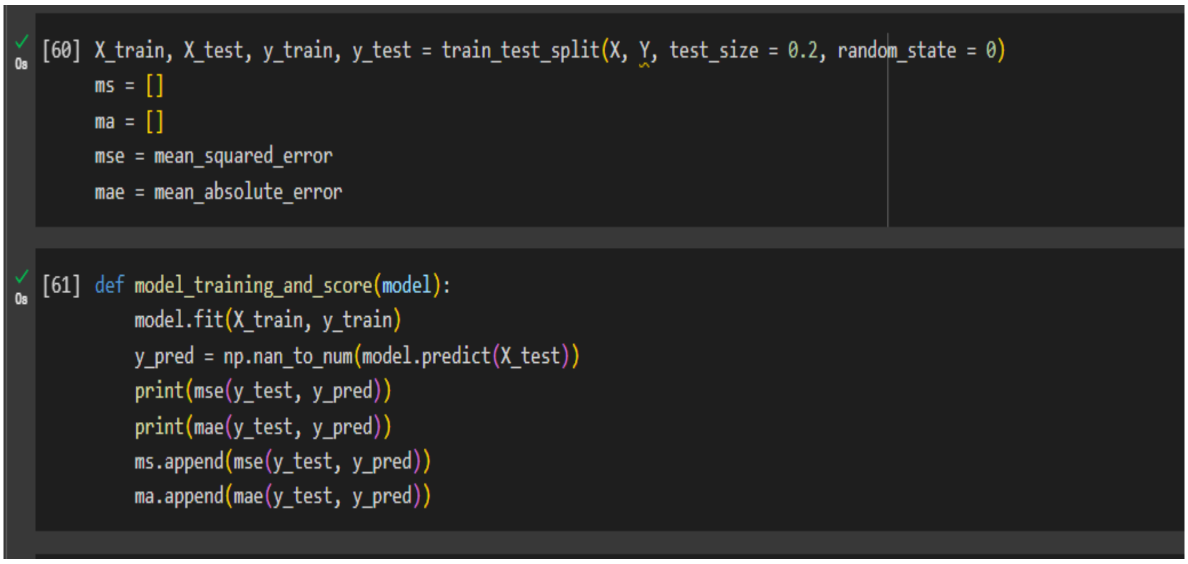
## Some Common Techniques :

### Import libraries



```
[67] import matplotlib.pyplot as plt
     from sklearn.metrics import mean_squared_error, mean_absolute_error
     from sklearn.linear_model import Lasso, SGDRegressor, Ridge
     from sklearn.svm import SVR
     from sklearn.preprocessing import StandardScaler
     from sklearn.ensemble import RandomForestRegressor
     from sklearn.gaussian_process import GaussianProcessRegressor
     from sklearn.tree import DecisionTreeRegressor
     from sklearn.neighbors import KNeighborsRegressor
     from sklearn.neighbors import RadiusNeighborsRegressor
     from sklearn.model_selection import train_test_split
     import seaborn as sns
```

### Model Training

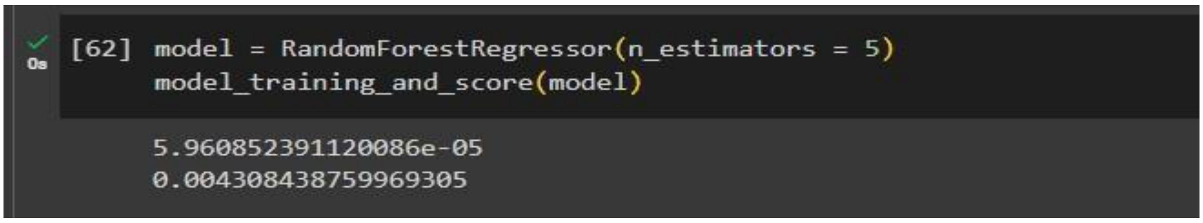


```
[60] X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
     ms = []
     ma = []
     mse = mean_squared_error
     mae = mean_absolute_error
```

```
[61] def model_training_and_score(model):
         model.fit(X_train, y_train)
         y_pred = np.nan_to_num(model.predict(X_test))
         print(mse(y_test, y_pred))
         print(mae(y_test, y_pred))
         ms.append(mse(y_test, y_pred))
         ma.append(mae(y_test, y_pred))
```

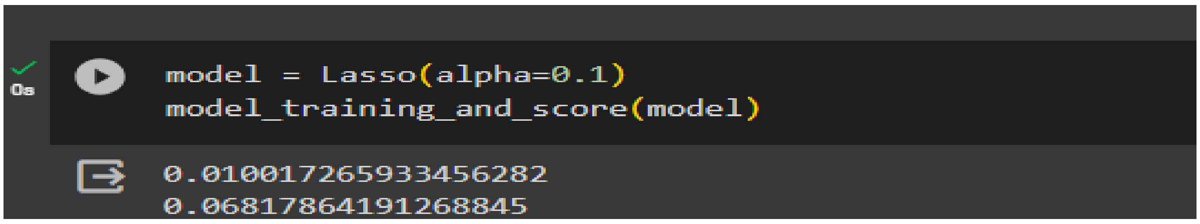
## MODEL EVALUATION

### 1.Random Forest



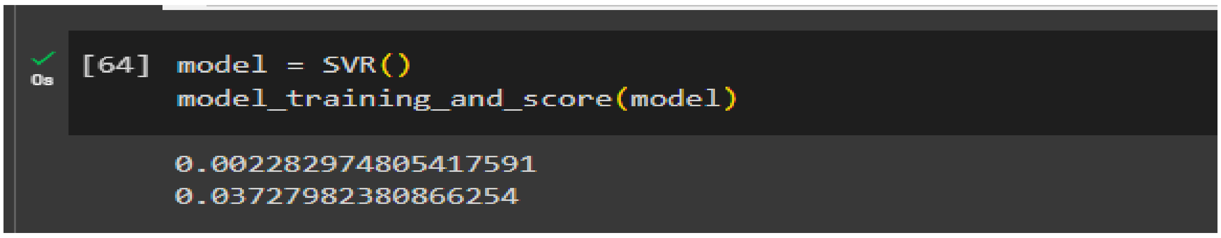
```
[62] model = RandomForestRegressor(n_estimators = 5)
     model_training_and_score(model)

     5.960852391120086e-05
     0.004308438759969305
```

### 2. Lasso Regression



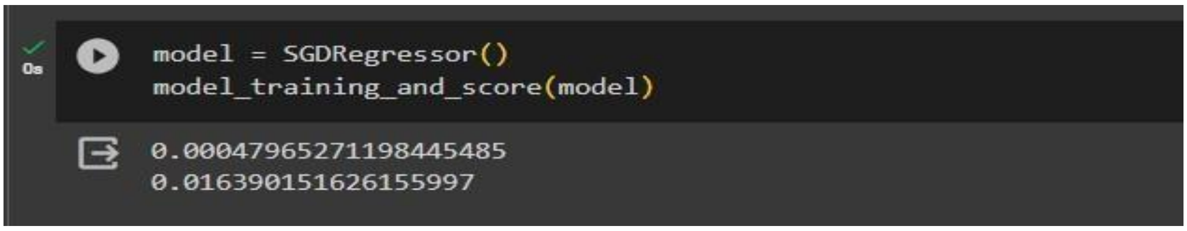
```
model = Lasso(alpha=0.1)
model_training_and_score(model)

0.010017265933456282
0.06817864191268845
```

### 3. Support Vector Regressor



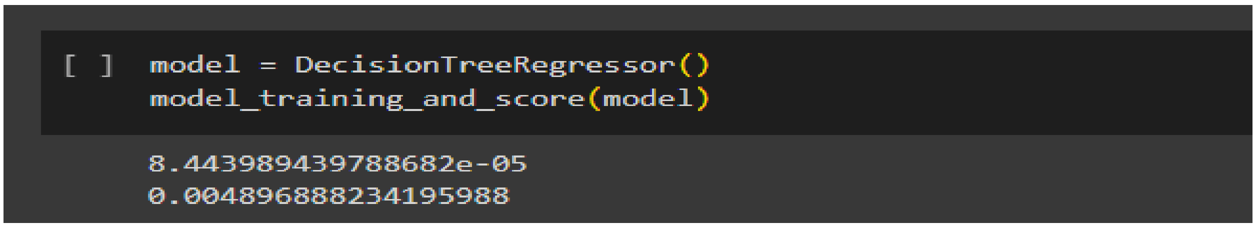
```
[64] model = SVR()
     model_training_and_score(model)

     0.002282974805417591
     0.03727982380866254
```

### 4. Stochastic Gradient Descent



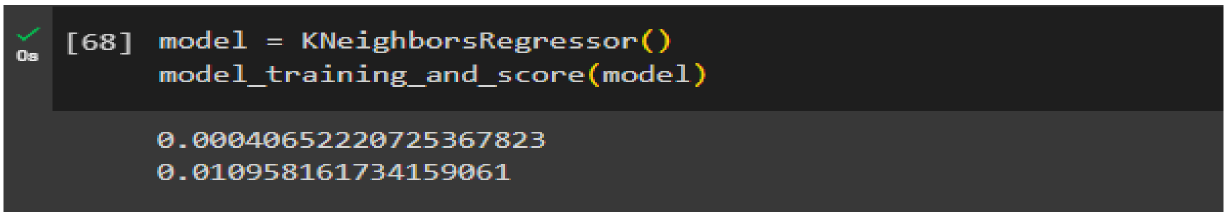
```
model = SGDRegressor()
model_training_and_score(model)

0.00047965271198445485
0.016390151626155997
```

## 5. Decision Tree Regressor



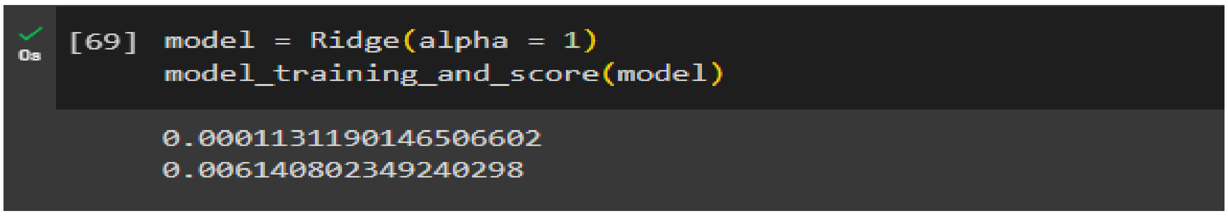
```
[ ]  model = DecisionTreeRegressor()
     model_training_and_score(model)

     8.443989439788682e-05
     0.004896888234195988
```

## 6. K Neighbors Regressor



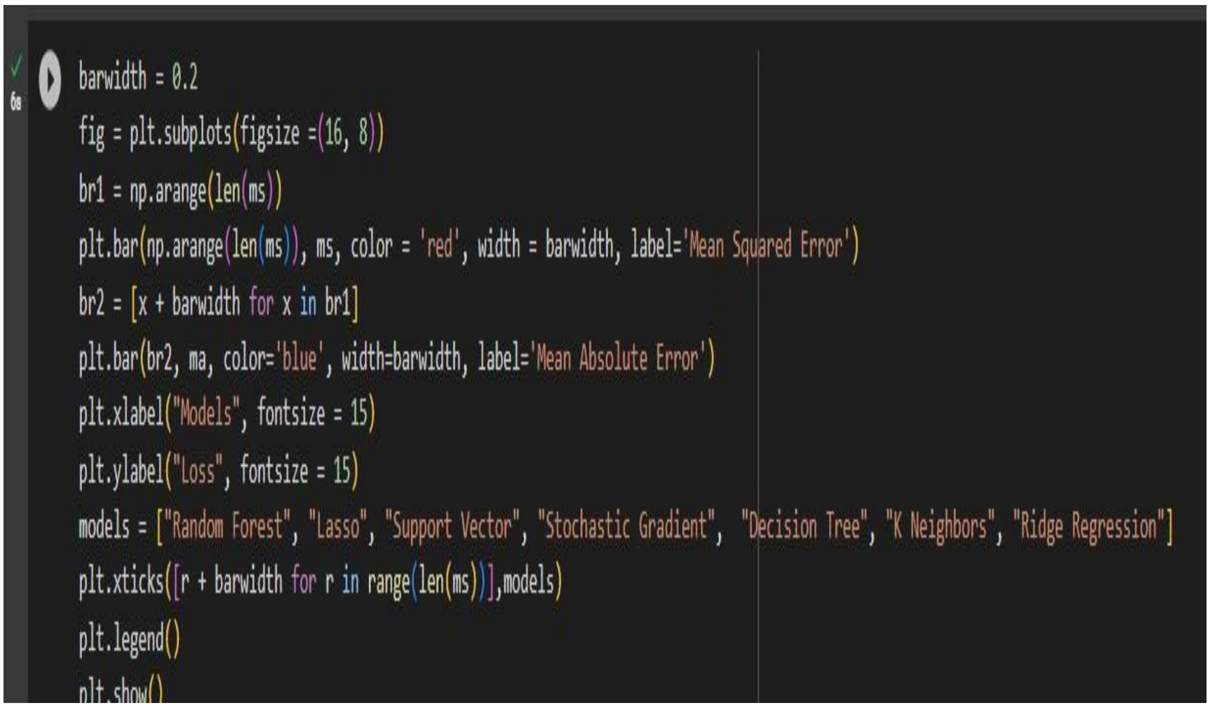
```
[68]  model = KNeighborsRegressor()
      model_training_and_score(model)

      0.00040652220725367823
      0.010958161734159061
```

## 7. Ridge Regression

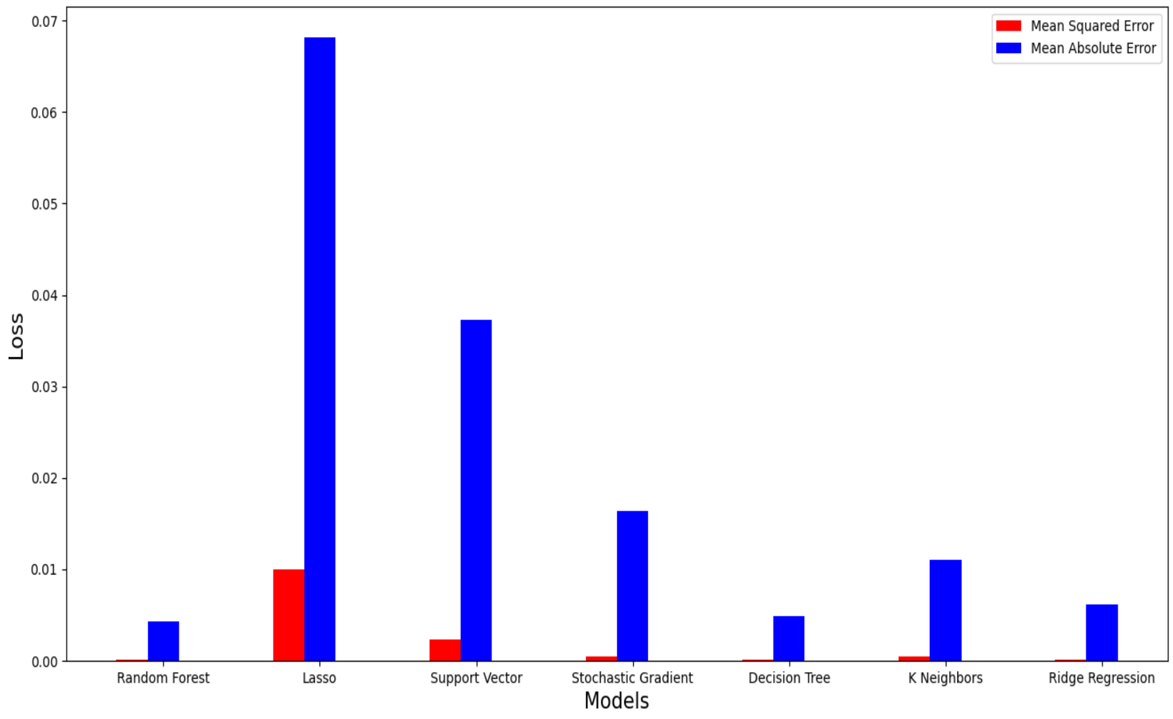


```
[69]  model = Ridge(alpha = 1)
      model_training_and_score(model)

      0.00011311901465066O2
      0.006140802349240298
```

## Plotting Losses of different models



```
barwidth = 0.2
fig = plt.subplots(figsize =(16, 8))
br1 = np.arange(len(ms))
plt.bar(np.arange(len(ms)), ms, color = 'red', width = barwidth, label='Mean Squared Error')
br2 = [x + barwidth for x in br1]
plt.bar(br2, ma, color='blue', width=barwidth, label='Mean Absolute Error')
plt.xlabel("Models", fontsize = 15)
plt.ylabel("Loss", fontsize = 15)
models = ["Random Forest", "Lasso", "Support Vector", "Stochastic Gradient",  "Decision Tree", "K Neighbors", "Ridge Regression"]
plt.xticks([r + barwidth for r in range(len(ms))],models)
plt.legend()
plt.show()
```



**CONCLUSION:**

- In conclusion, while data science techniques can provide valuableinsights into stock price movements, the inherent complexity anduncertainty of financial markets mean that predictions should be used as one of several tools in the decision-making process. Additionally, ethical considerations, transparency, and a deep understanding of financial markets are essential when applying data science to stock price prediction.
- stock price prediction is a complex and challenging task that requires a combination of domain expertise, data manipulation, and machine learning techniques. While there is no foolproof method to accurately forecast stock prices due to the influence of numerous external factors and market dynamics, machinelearning algorithms, particularly deep learning models like LSTM, have shown promise in capturing temporal pattern.
- However, predicting stock prices accurately remains challengingdue to the multitude of external factors. While predictive modelscan provide valuable insights, they should be used in conjunction with a comprehensive understanding of market dynamics, risk management, and diversification strategies to inform investment decisions.

REFERENCES:

- https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571

- https://towardsdatascience.com/predicting-stock-prices-using-a-keras-lstm-model-4225457f0233

- https://neptune.ai/blog/predicting-stock-prices-using-machine-learning

- https://www.simplilearn.com/tutorials/machine-learning-tutorial/stock-price-prediction-using-machine-learning