Kathmandu University
Department of Mathematics

Dhulikhel, Kavre

A Project Report

On

“NMB BANK SHAREMARKET DATA ANALYSIS”

[Subject Code: DSMA 113]

Submission Date: 24/02/2025

Acknowledgement

We would like to express our heartfelt gratitude to the Department of Mathematics project community who provides us with this opportunity to do our project *“**NMB BANK SHAREMARKET DATA ANALYSIS**”*, which has enabled us to explore innovative ways of using multivariate regression analysis for the study of share market and make prediction of future closing prices.We sincerely thank **Assistant Prof. Narayan Sapkota**, our project supervisor, for his invaluable guidance, support, and encouragement throughout this project. His expertise and feedback greatly enhanced our work.

This project would not have been possible without the collective efforts and support of each individual in this group. I am grateful for their contributions and am honored to have worked alongside such inspiring individuals.

Lastly, we would like to extend our appreciation to our dear senior **Ashim Shrestha** for his valuable guidance and support.

Table Of Content

Introduction

Multivariable regression, also known as multiple linear regression, is a statistical technique used to model the relationship between a dependent variable and multiple independent variables. In the context of stock market analysis, this method is valuable for understanding how various factors influence stock prices.

In this report, we will apply multi-variable regression to the NMB bank stock dataset to develop a predictive model for stock price estimation. Multivariable regression is widely used in financial analytics for trend forecasting, risk assessment, and investment decision-making.

The analysis will include data preprocessing, model implementation, evaluation metrics, and interpretation of results. Multivariable regression is widely used in financial analytics for trend forecasting, risk assessment, and investment decision-making. With the help of Python and its powerful libraries, we can efficiently analyze NMB's stock data and extract meaningful insights.

The following sections will guide you through dataset selection, preprocessing steps, model development, and performance evaluation, showcasing the practical applications of multivariable regression in stock market analysis.

Objective

The primary objectives of this study are:

- To understand the fundamental principles of Multivariate Linear Regression.
- To analyze historical stock market data using MLR.
- To build a predictive model for stock closing prices using multiple financial indicators.
- To evaluate model performance using statistical metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2).
- To visualize actual vs. predicted stock prices and make future predictions.

Motivation

The motivation for us to do this project is to challenge the everchanging stock market which is changing every minute based on the various happening that is going around in the world.As, the stock market is a dynamic and complex financial system that influences economies worldwide, predicting stock prices accurately can provide significant advantages to investors, traders, and financial institutions, helping them make informed decisions and minimize risks. Traditional methods of stock analysis rely heavily on fundamental and technical indicators, but with the rise of data-driven approaches, machine learning and statistical models have opened new possibilities for market forecasting. Predicting the stock market is not for the one without any dedication. It's a challenge that demands rigorous analysis and constant learning. However, we are not fortune tellers; we are students of the market, researchers seeking patterns, and strategists crafting informed decisions.

Background Theory: Multivariate Linear Regression

Multivariate Linear Regression (MLR) is a statistical technique used to model the relationship between multiple independent variables and a single dependent variable. It extends simple linear regression by incorporating multiple predictors to better capture complex relationships in the data.

$$\mathbf{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n + \epsilon$$

where:

- **Y** is the dependent variable (target output).
- $X_1, X_2, \ldots, X_n$ are the independent variables(predictors) .
- β_0 is the intercept.
- $\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients that represent the impact of each independent variable.
- ϵ is the error term, capturing variations not explained by the model.

Assumptions of MLR:

1. Linearity : The relationship between dependent and independent variables is linear.
2. Independence : Observations are independent of each other.
3. No Multicollinearity : Independent variables should not be highly correlated.
4. Homoscedasticity :Constant variance of residuals.
5. Normality : The residuals should be normally distributed.

Model Evaluation Metrics:

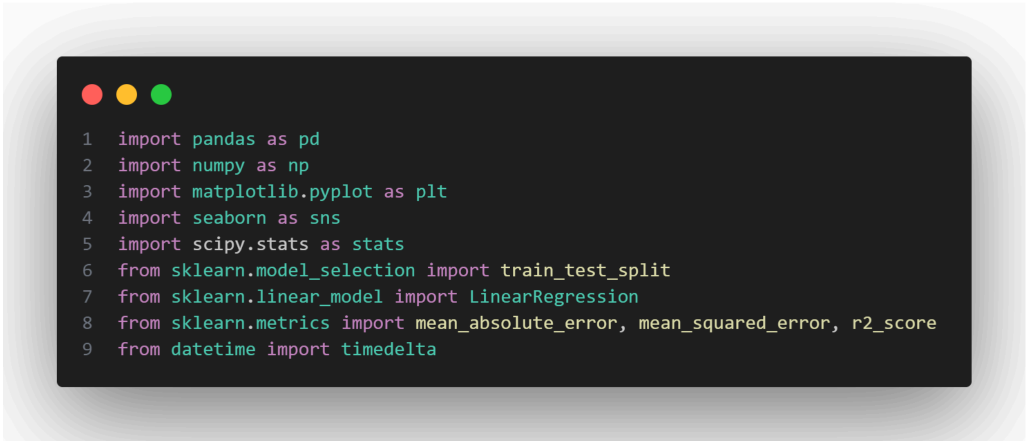
- R-squared (R^2) :Measures how well the independent variables explain the variance in the dependent variable.
- Mean Squared Error (MSE) :Indicates the average squared differences between actual and predicted values.
- Mean Absolute Error (MAE) :Measures the absolute differences between actual and predicted values.

MLR is widely used in fields such as **finance, economics, healthcare, and marketing** for predictive analysis and decision-making. In Python, it can be implemented using the sklearn library, making it a powerful tool for real-world applications.

Methodology

This section outlines the methodological approach taken for data analysis, including Data Collection and Preprocessing Exploratory Data Analysis (EDA), Descriptive Analysis, and Regression Analysis.

We imported the following libraries:

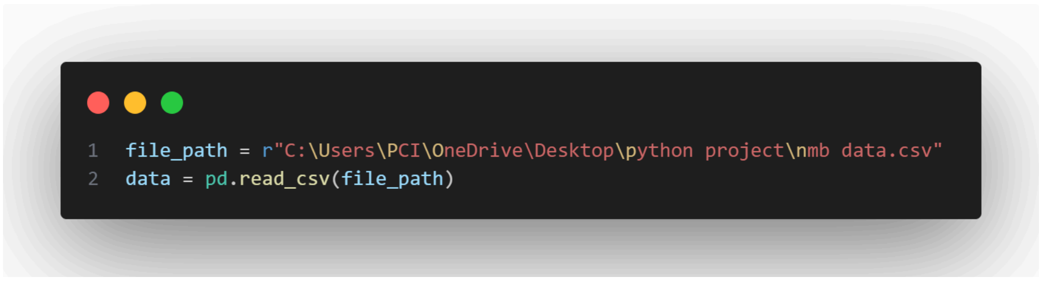


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from datetime import timedelta
```

1. Data Collection and Preprocessing:

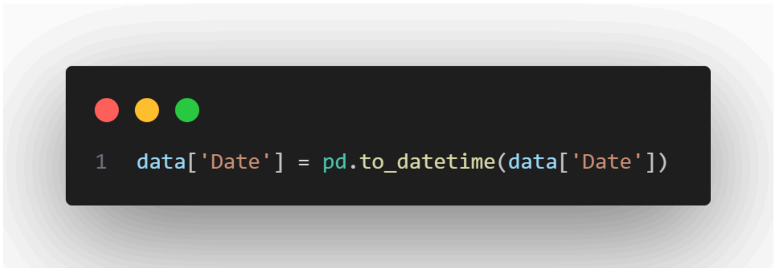
The dataset used for this analysis was sourced from ***"nmb data.csv"***. We created this file by collecting the data from online resources. The data was loaded into Python using the Pandas library. As part of preprocessing, the Date column was converted to a datetime format to facilitate time-based analysis. Missing values and outliers were identified and handled appropriately to maintain data integrity.

Loading Stock Data from CSV file



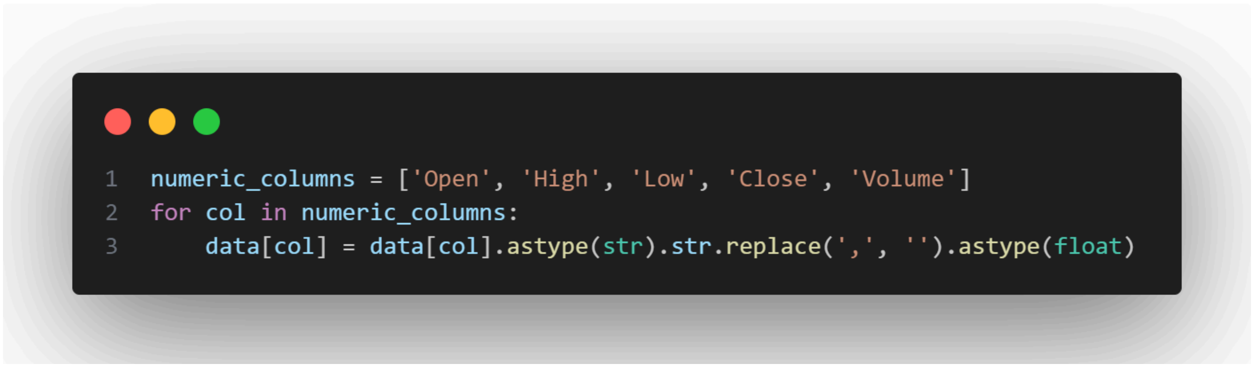
```python
file_path = r"C:\Users\PCI\OneDrive\Desktop\python project\nmb data.csv"
data = pd.read_csv(file_path)
```

Making sure 'DATE' column is in datetime format



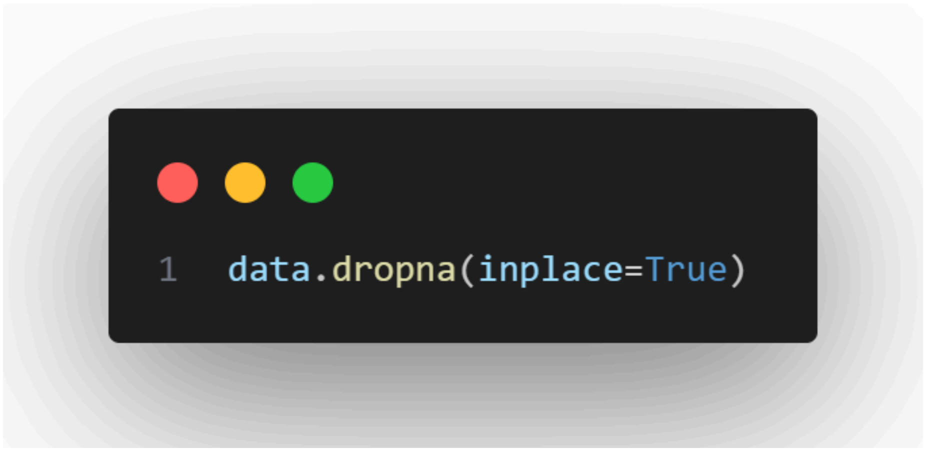
```python
data['Date'] = pd.to_datetime(data['Date'])
```

Converting numeric columns to float and removing unnecessary commas



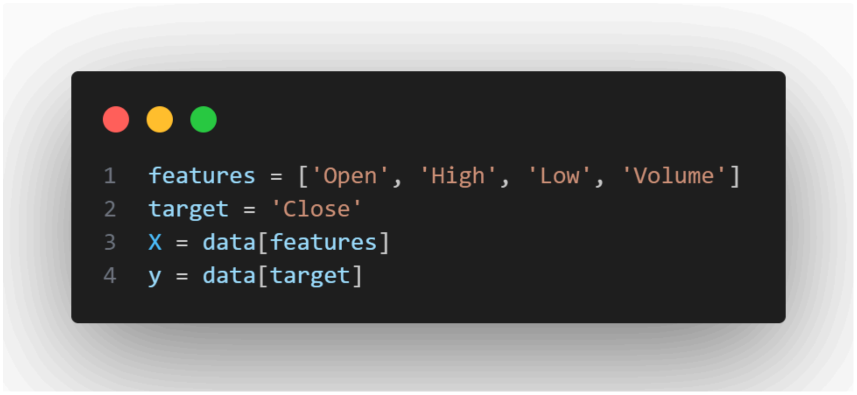
```python
numeric_columns = ['Open', 'High', 'Low', 'Close', 'Volume']
for col in numeric_columns:
    data[col] = data[col].astype(str).str.replace(',', '').astype(float)
```

Deleting rows with missing values


```
data.dropna(inplace=True)
```

2. Feature Selection and Data Splitting :

Relevant independent variables ("X") were selected based on their potential impact on the dependent variable ("y"). The dataset was then split into training and testing sets using an 80-20 ratio to ensure effective model training and evaluation. This was achieved using the train_test_split function from the sklearn.model_selection module.


```
features = ['Open', 'High', 'Low', 'Volume']
target = 'Close'
X = data[features]
y = data[target]
```

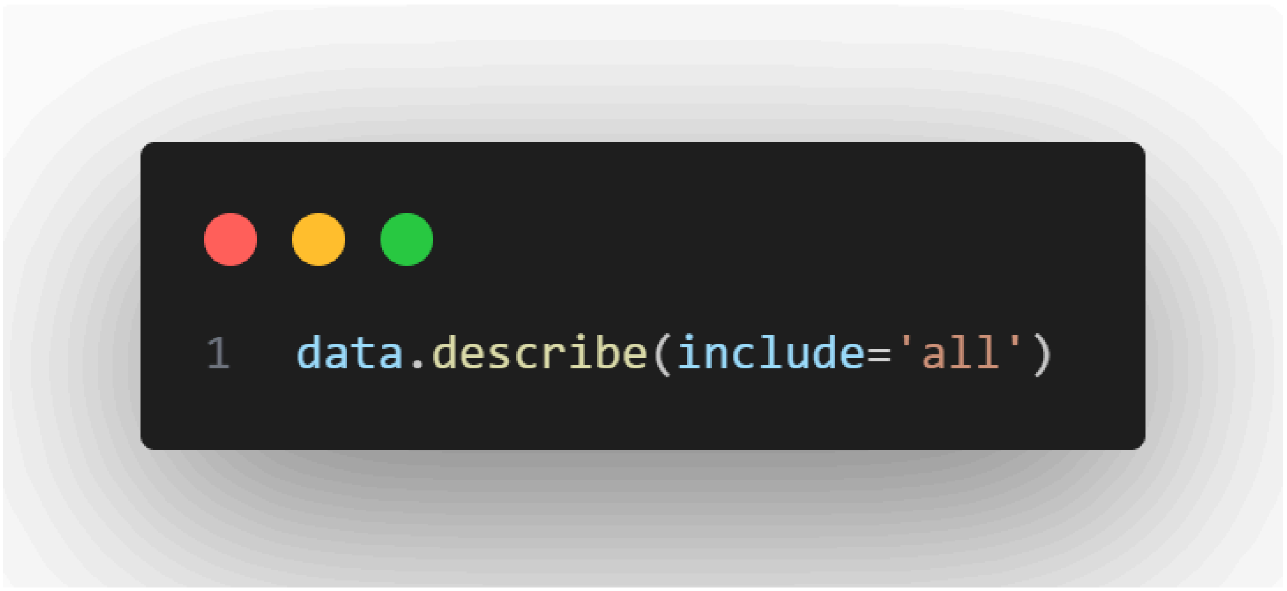
Defining features as independent variables and target as dependent variable

3. Descriptive Analysis

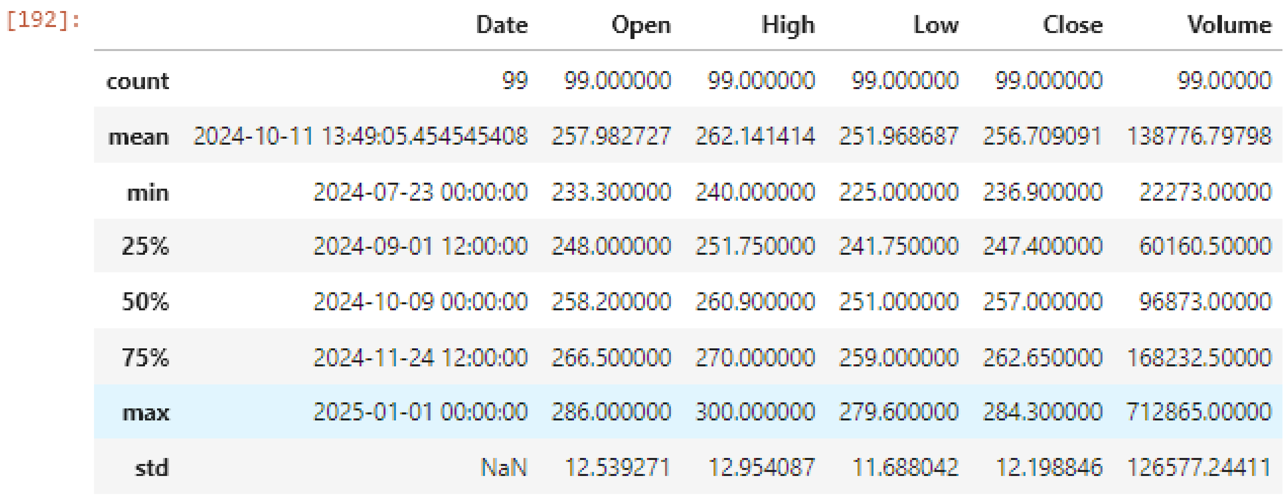
A descriptive analysis of the dataset was conducted using the **data.describe(include='all')** function, which provided key statistical insights:

- **Count:** The dataset contains 1,719 observations for each variable.
- **Mean Values:** The average values for key stock indicators were as follows:
  - Open Price: 257.98
  - High Price: 262.14
  - Low Price: 251.97
  - Close Price: 256.71
  - Trading Volume: 138,776.80
- **Minimum and Maximum Values:**
  - The lowest recorded stock price (Open) was 233.80, while the highest was 286.00.
  - The highest recorded trading volume was 712,865, while the lowest was 22,773.
- **Percentile Distribution:**
  - 25% of the data had an Open price below 248.00, while 75% was below 266.50.
  - The median (50%) Open price was 258.00, indicating a central tendency.
- **Standard Deviation:**The standard deviation for stock prices ranged as follows:
  - Open: 12.53
  - High: 12.95
  - Low: 11.68
  - Close: 12.19

  The trading volume had a standard deviation of 126,577.24, suggesting significant fluctuations over time.



```python
data.describe(include='all')
```



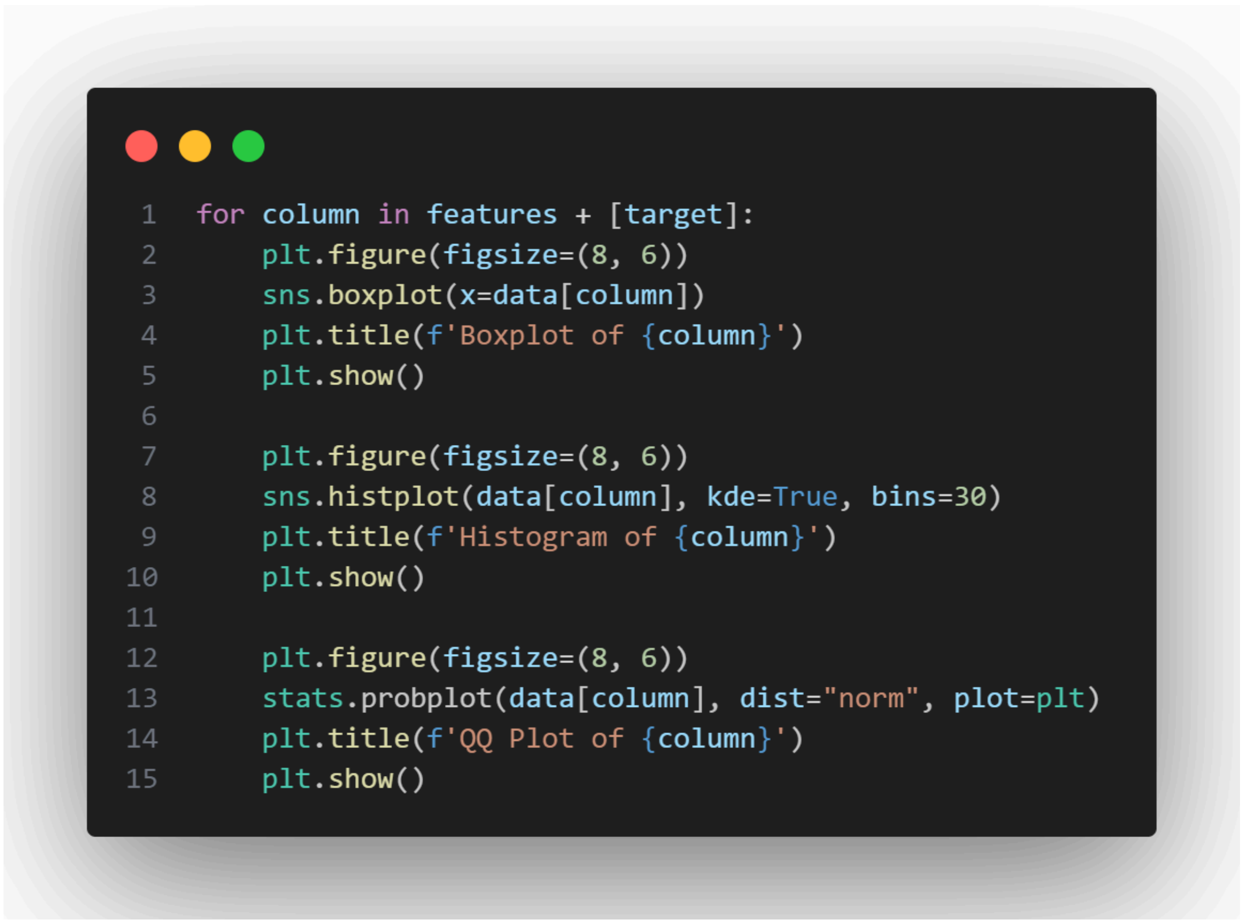
[192]:

	Date	Open	High	Low	Close	Volume
count	99	99.000000	99.000000	99.000000	99.000000	99.00000
mean	2024-10-11 13:49:05.454545408	257.982727	262.141414	251.968687	256.709091	138776.79798
min	2024-07-23 00:00:00	233.300000	240.000000	225.000000	236.900000	22273.00000
25%	2024-09-01 12:00:00	248.000000	251.750000	241.750000	247.400000	60160.50000
50%	2024-10-09 00:00:00	258.200000	260.900000	251.000000	257.000000	96873.00000
75%	2024-11-24 12:00:00	266.500000	270.000000	259.000000	262.650000	168232.50000
max	2025-01-01 00:00:00	286.000000	300.000000	279.600000	284.300000	712865.00000
std	NaN	12.539271	12.954087	11.688042	12.198846	126577.24411

Fig: Descriptive statistics

4. Exploratory Data Analysis (EDA)

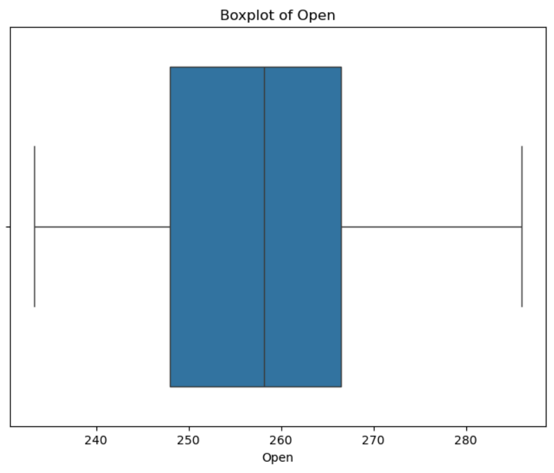
EDA was conducted to understand the dataset's structure and characteristics before performing further analysis. The following techniques were utilized:

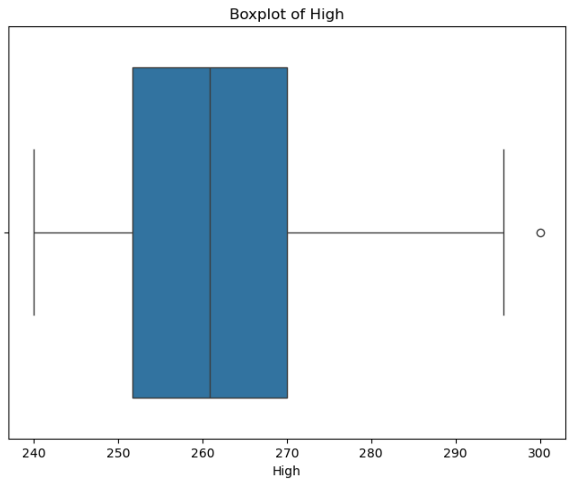


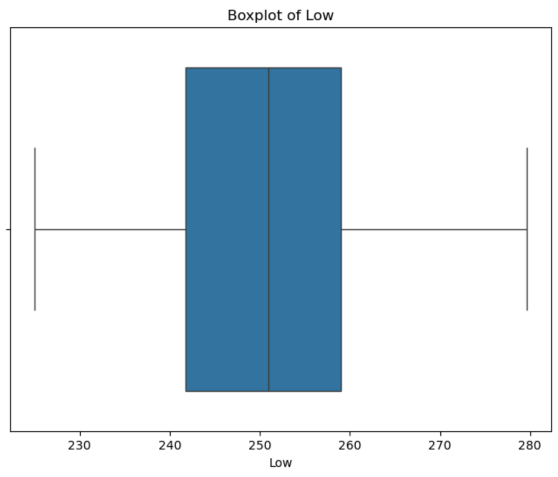
```python
for column in features + [target]:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=data[column])
    plt.title(f'Boxplot of {column}')
    plt.show()

    plt.figure(figsize=(8, 6))
    sns.histplot(data[column], kde=True, bins=30)
    plt.title(f'Histogram of {column}')
    plt.show()

    plt.figure(figsize=(8, 6))
    stats.probplot(data[column], dist="norm", plot=plt)
    plt.title(f'QQ Plot of {column}')
    plt.show()
```

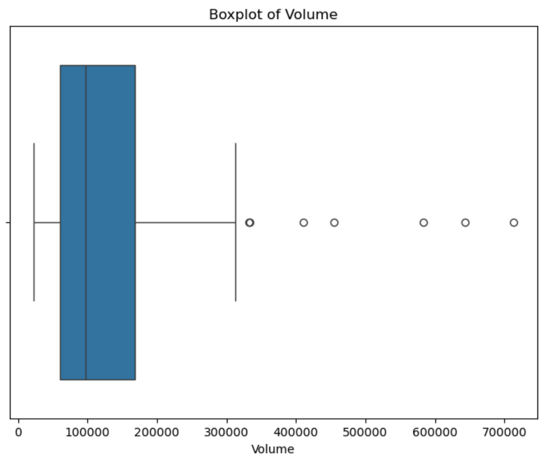
Code for BOX PLOT, HISTOGRAM and Q-Q PLOT

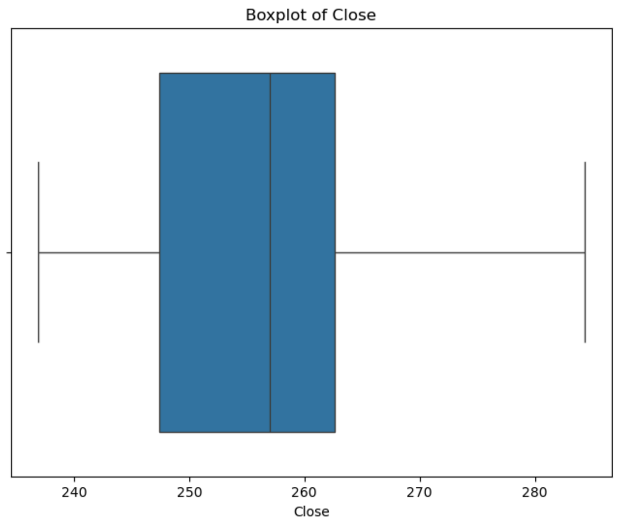
- **Box Plot**: Applied to detect outliers and analyze the distribution of numerical variables.



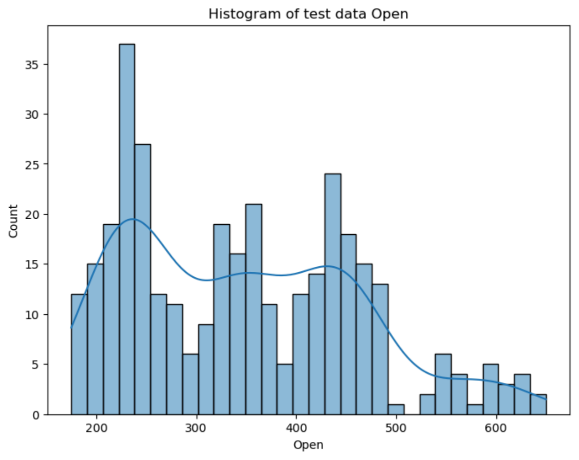


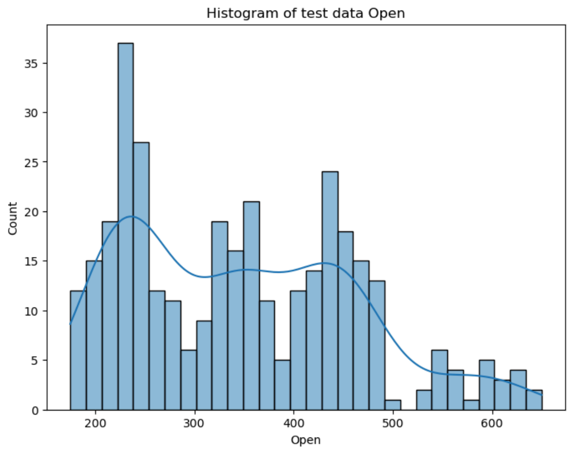


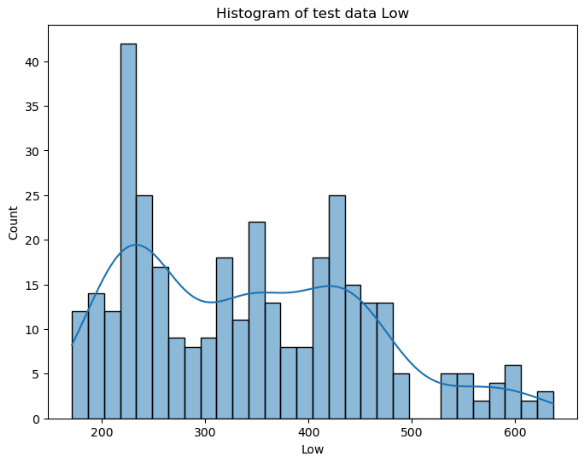


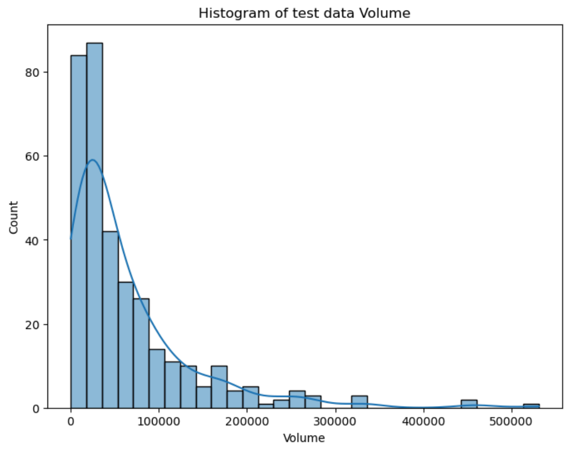


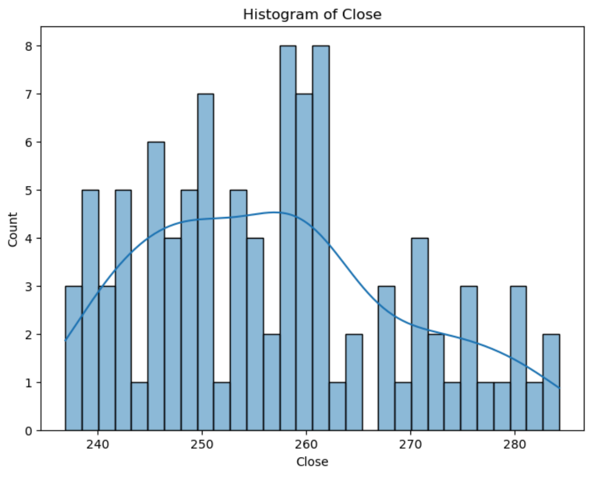
- **Histogram**: Used to display the frequency distribution of data, providing insights into its shape.





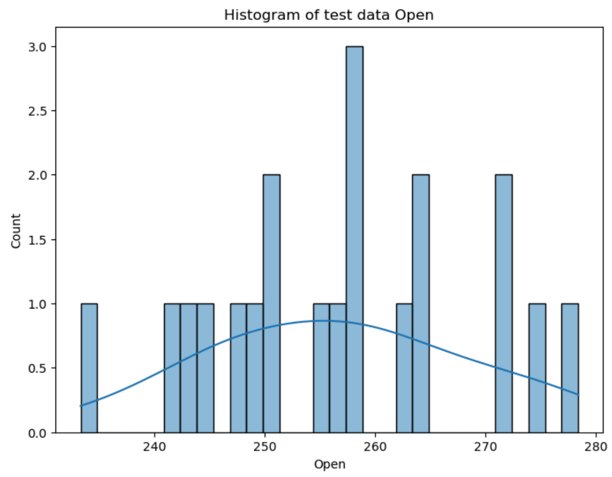


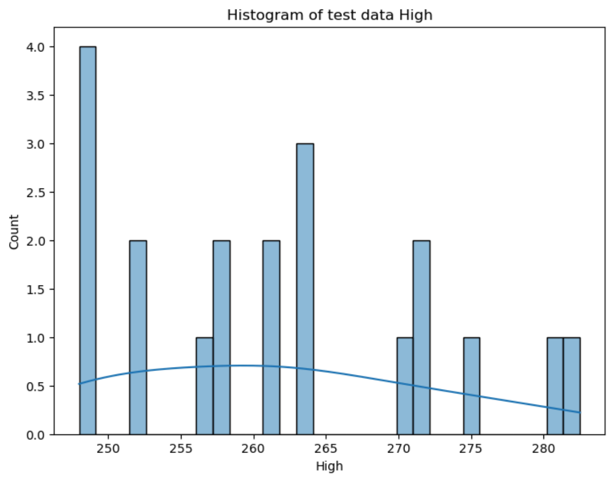


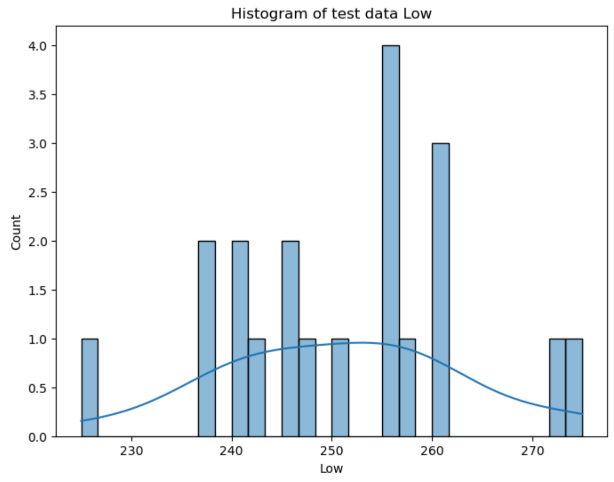


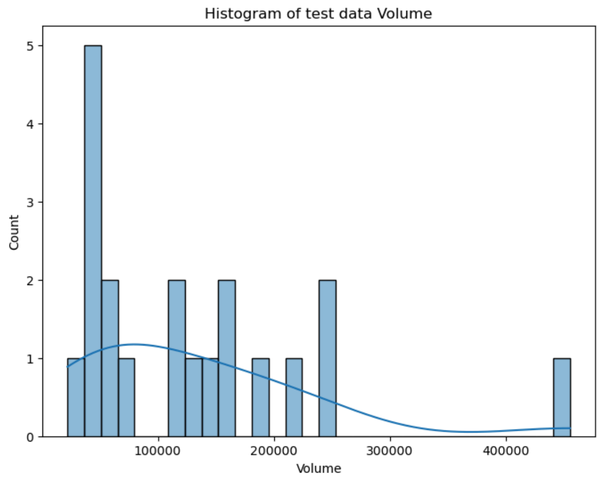
These are the plots of train data after the TRAIN-SPLIT test (using 80% train-20% test).

And below are the plots of test data:

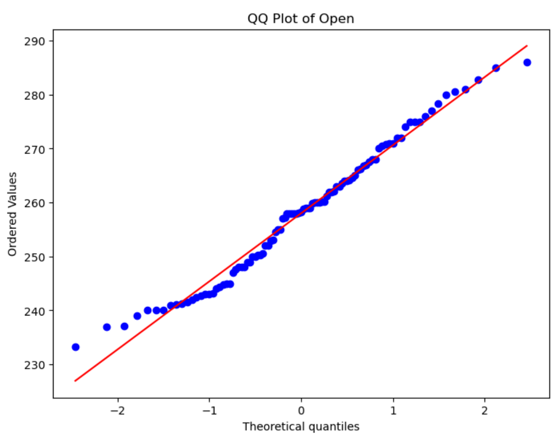


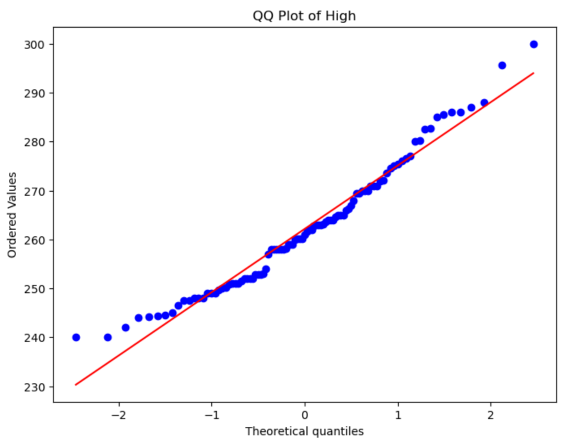


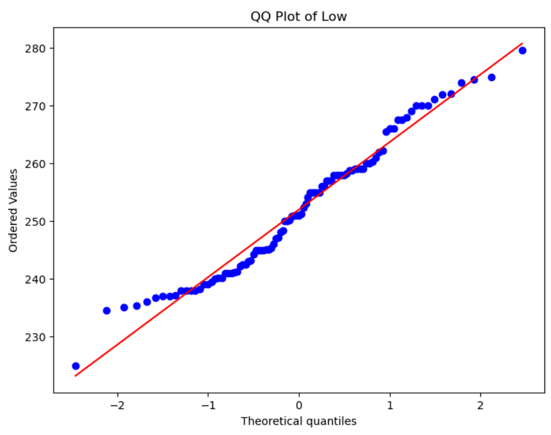


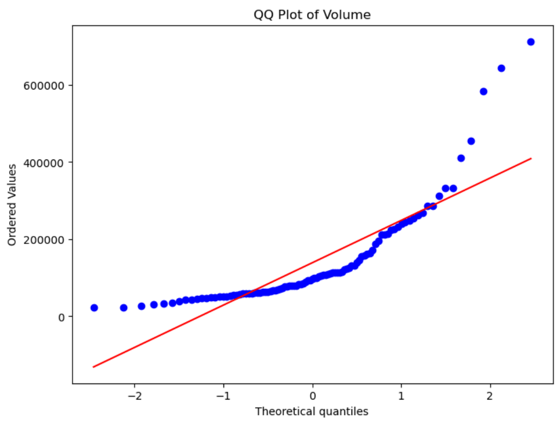


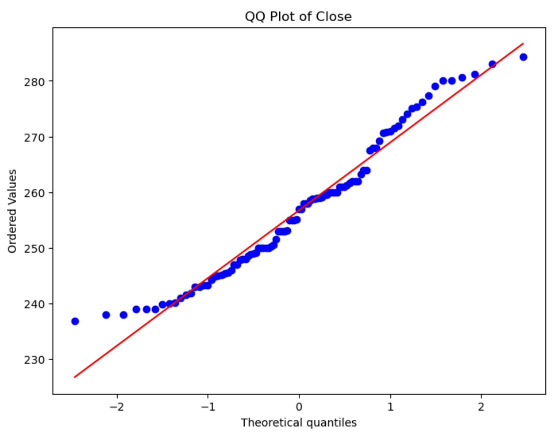
- **Q-Q Plot**: Employed to assess whether the data follows a normal distribution.



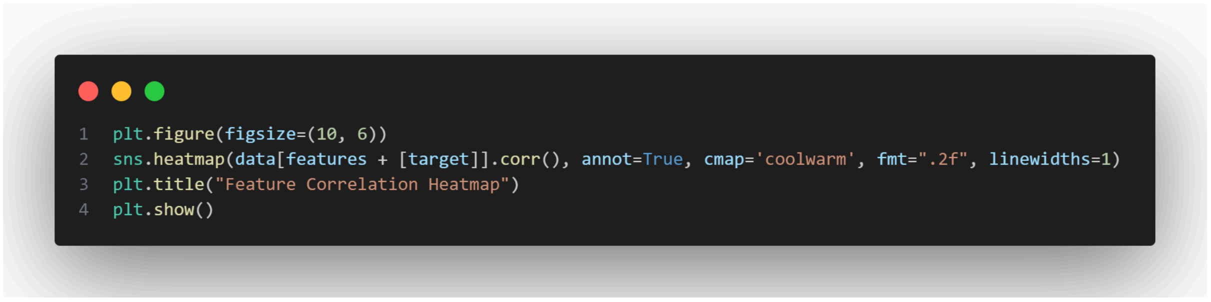




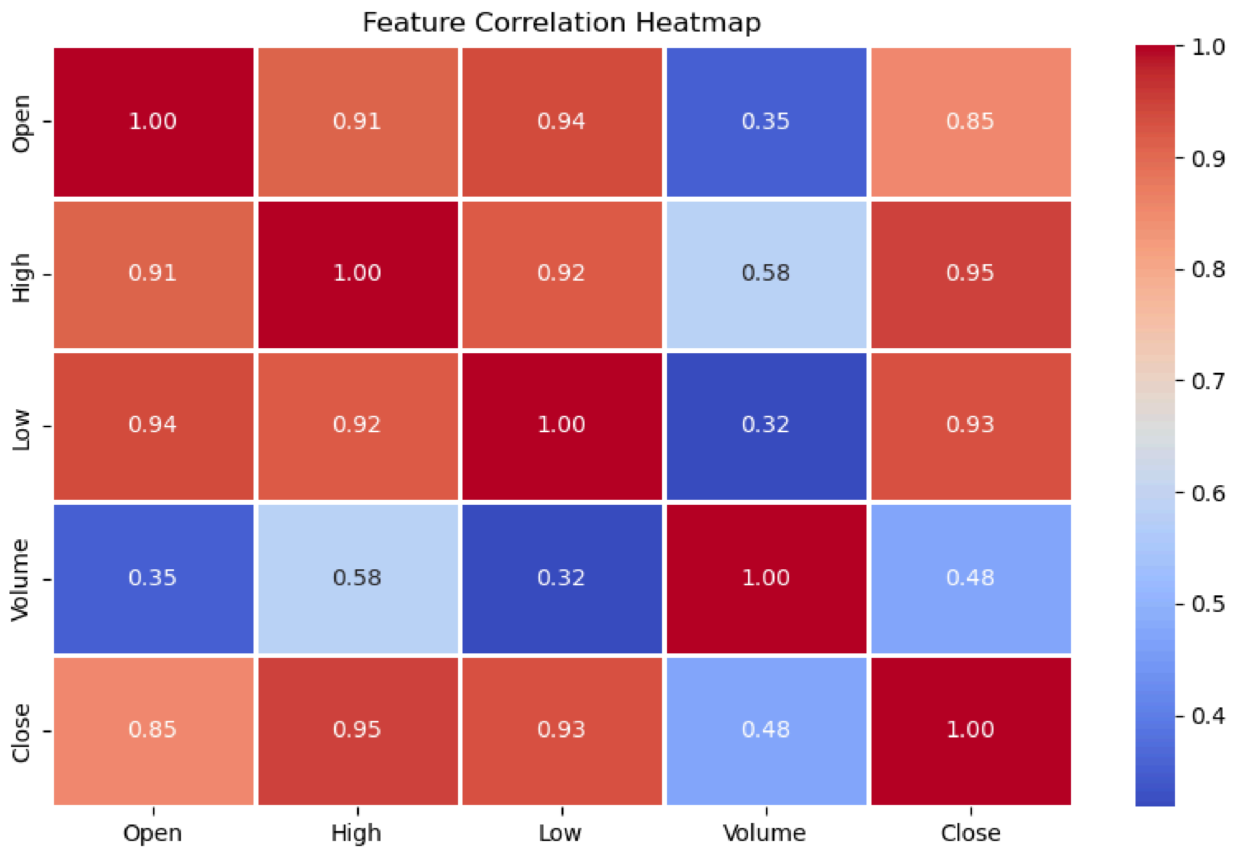




- **Heatmap**: Generated to visualize correlations between variables, aiding in feature selection for regression analysis.

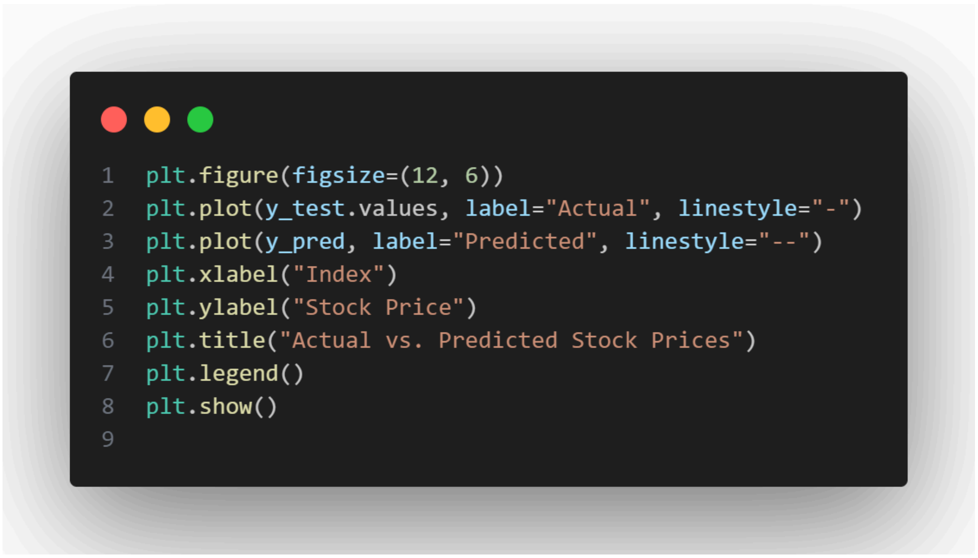


```python
plt.figure(figsize=(10, 6))
sns.heatmap(data[features + [target]].corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=1)
plt.title("Feature Correlation Heatmap")
plt.show()
```

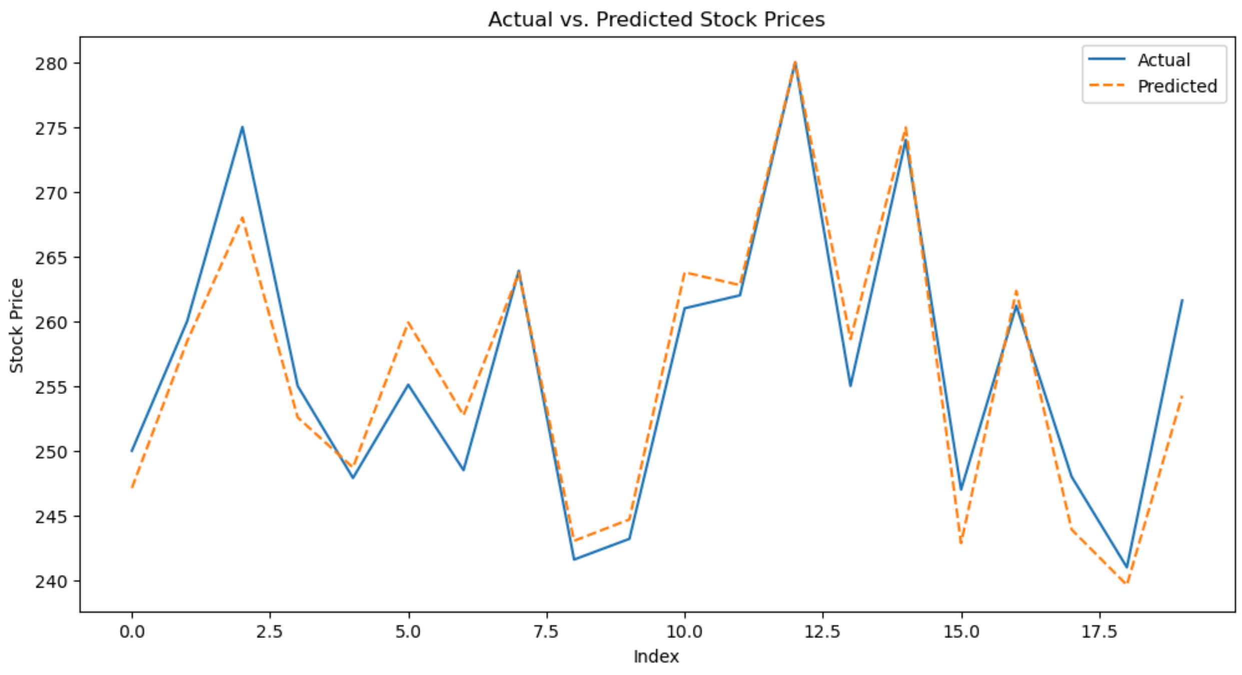


	Open	High	Low	Volume	Close
Open	1.00	0.91	0.94	0.35	0.85
High	0.91	1.00	0.92	0.58	0.95
Low	0.94	0.92	1.00	0.32	0.93
Volume	0.35	0.58	0.32	1.00	0.48
Close	0.85	0.95	0.93	0.48	1.00

- **Line Plot**: Used to visualize trends over time, particularly for time-series data.



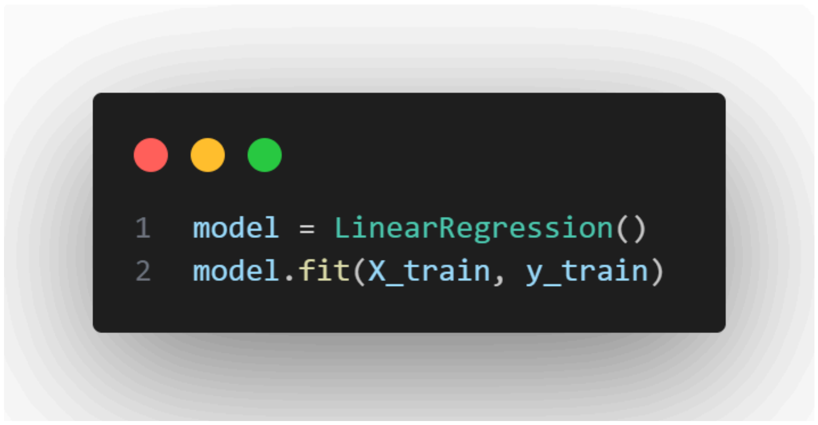
```python
plt.figure(figsize=(12, 6))
plt.plot(y_test.values, label="Actual", linestyle="-")
plt.plot(y_pred, label="Predicted", linestyle="--")
plt.xlabel("Index")
plt.ylabel("Stock Price")
plt.title("Actual vs. Predicted Stock Prices")
plt.legend()
plt.show()

```



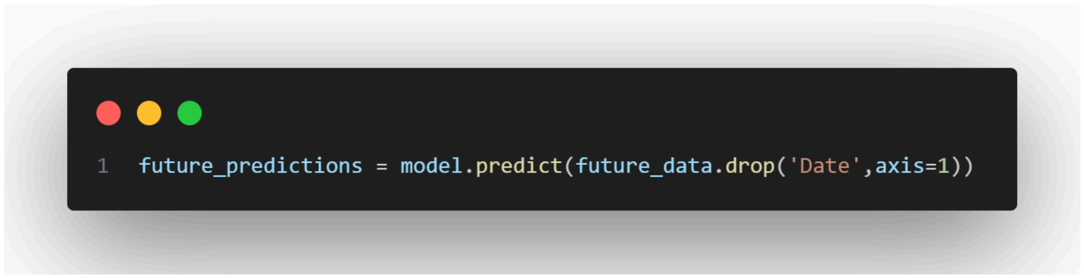
5. Regression Analysis:

- **Model Selection:**
    - ❖ A **Linear Regression Model** was chosen for this study due to its ability to quantify relationships between independent and dependent variables. The LinearRegression() function from sklearn.linear_model was used to implement the model.

- **Model Training and Prediction:**
    - ❖ The training dataset (X_train, y_train) was fed into the linear regression model using the .fit() method. Predictions were generated for the test dataset (X_test) using the .predict() method.

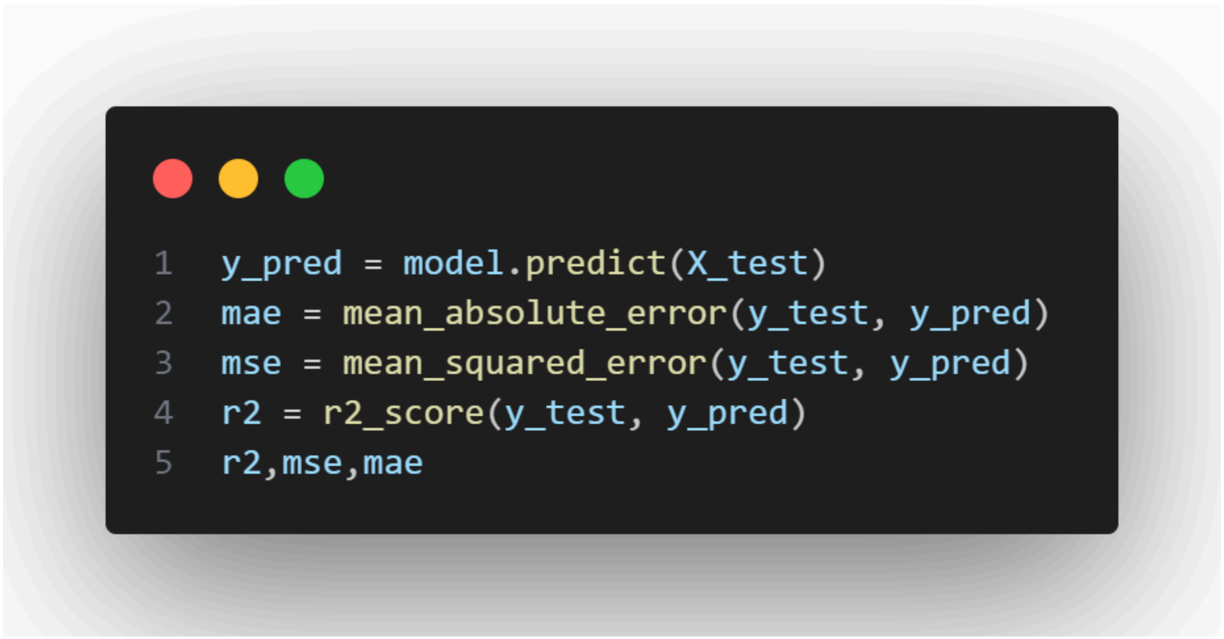


```
model = LinearRegression()
model.fit(X_train, y_train)
```



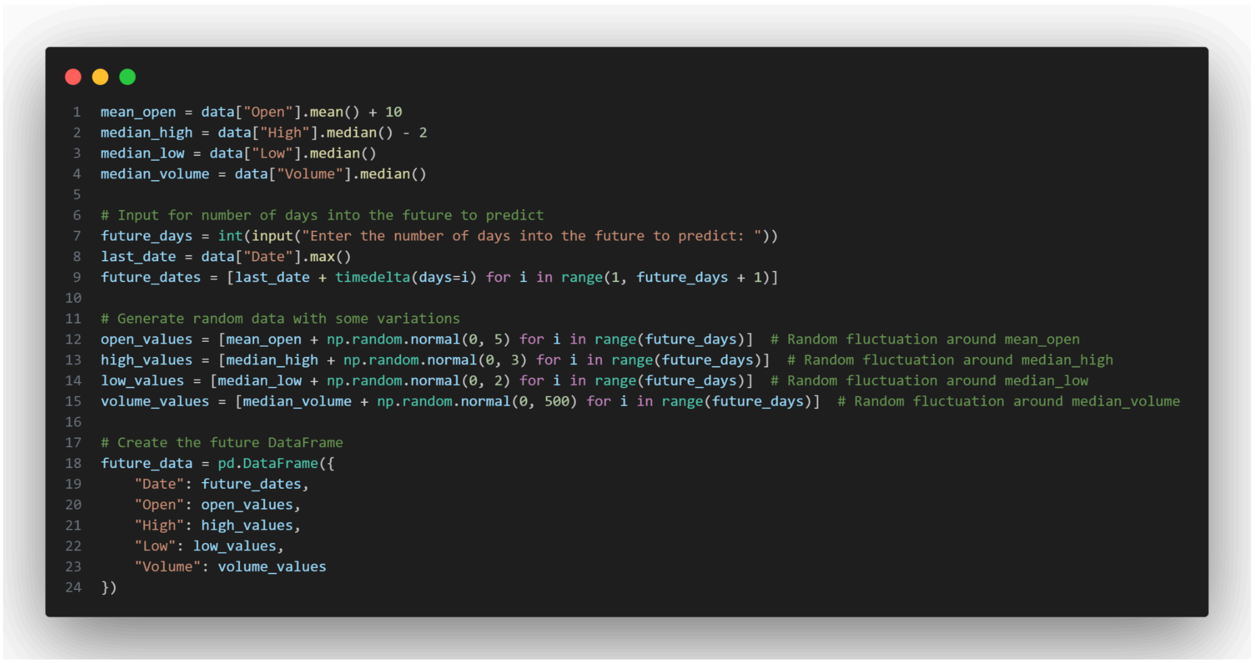
```
future_predictions = model.predict(future_data.drop('Date',axis=1))
```

- **Model Evaluation:** To assess the accuracy and performance of the regression model, the following metrics were calculated:
    I. **Mean Absolute Error (MAE):** Measures the average absolute differences between actual and predicted values.
    II. **Mean Squared Error (MSE):** Measures the average squared differences between actual and predicted values, giving more weight to larger errors.
    III. **R-squared (R^2):** Indicates how well the independent variables explain the variance in the dependent variable. Higher R^2 values suggest better model performance.



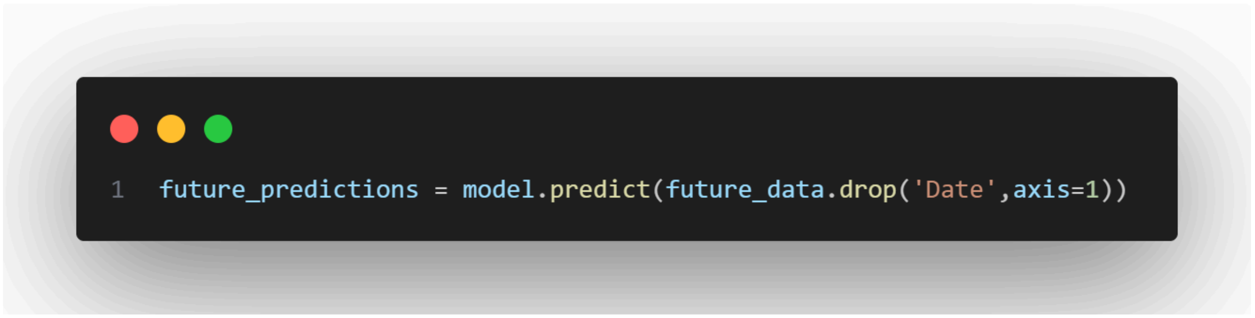
```
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
r2,mse,mae
```

OUTPUT: (0.9992232046500504, 10.187310002665804, 2.1964410521955267)



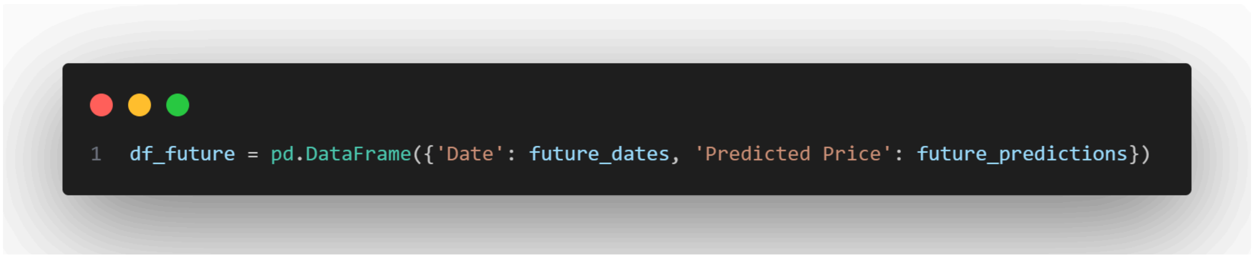
```python
mean_open = data["Open"].mean() + 10
median_high = data["High"].median() - 2
median_low = data["Low"].median()
median_volume = data["Volume"].median()

# Input for number of days into the future to predict
future_days = int(input("Enter the number of days into the future to predict: "))
last_date = data["Date"].max()
future_dates = [last_date + timedelta(days=i) for i in range(1, future_days + 1)]

# Generate random data with some variations
open_values = [mean_open + np.random.normal(0, 5) for i in range(future_days)]  # Random fluctuation around mean_open
high_values = [median_high + np.random.normal(0, 3) for i in range(future_days)]  # Random fluctuation around median_high
low_values = [median_low + np.random.normal(0, 2) for i in range(future_days)]  # Random fluctuation around median_low
volume_values = [median_volume + np.random.normal(0, 500) for i in range(future_days)]  # Random fluctuation around median_volume

# Create the future DataFrame
future_data = pd.DataFrame({
    "Date": future_dates,
    "Open": open_values,
    "High": high_values,
    "Low": low_values,
    "Volume": volume_values
})
```

Generating future data by increasing the values of features with expected mean

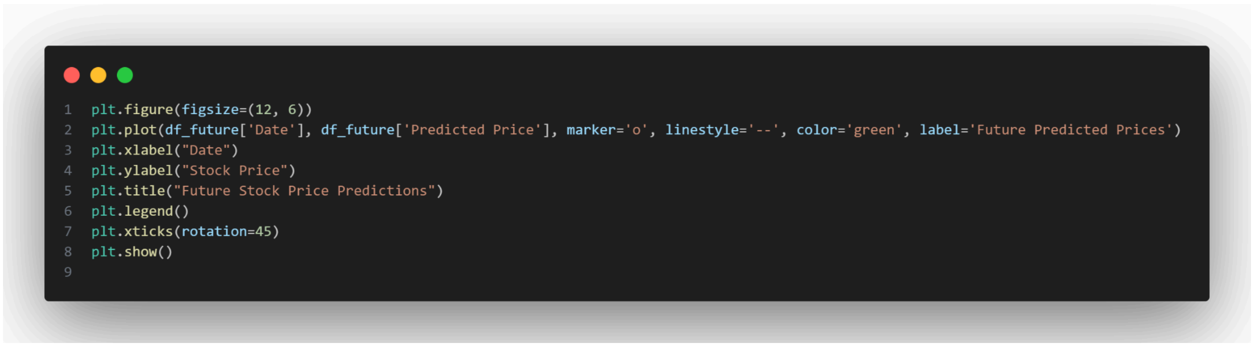


```python
future_predictions = model.predict(future_data.drop('Date',axis=1))
```

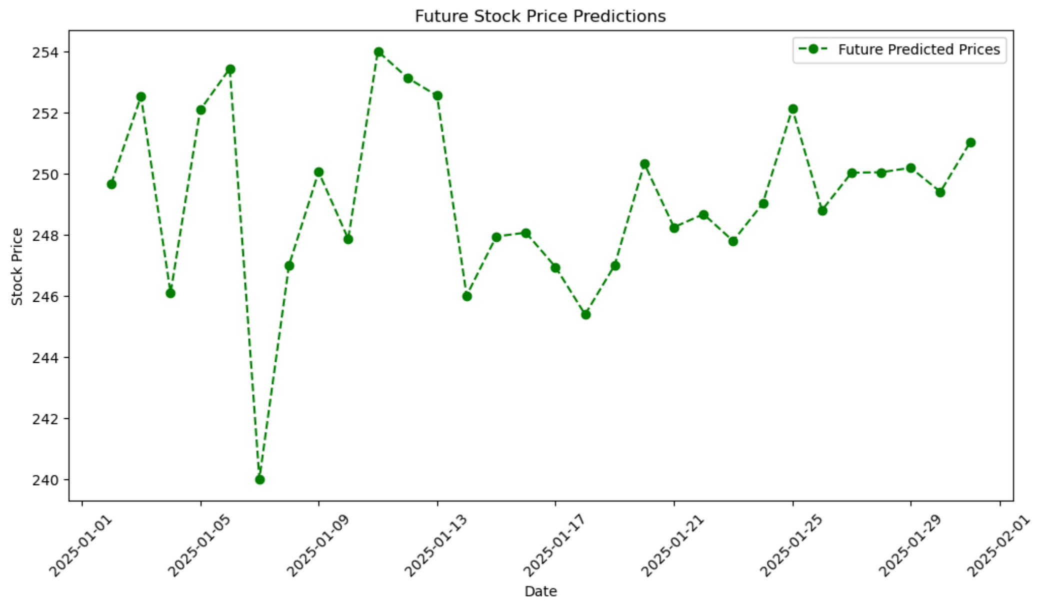
Adjusting future dates in the predicted data frame



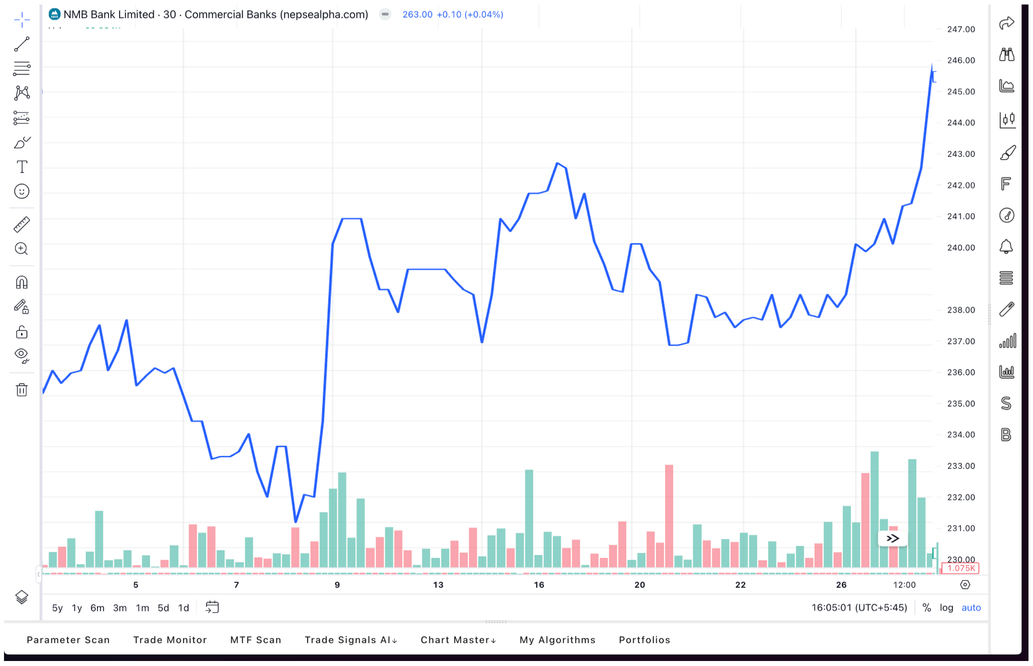
```python
df_future = pd.DataFrame({'Date': future_dates, 'Predicted Price': future_predictions})
```



```python
plt.figure(figsize=(12, 6))
plt.plot(df_future['Date'], df_future['Predicted Price'], marker='o', linestyle='--', color='green', label='Future Predicted Prices')
plt.xlabel("Date")
plt.ylabel("Stock Price")
plt.title("Future Stock Price Predictions")
plt.legend()
plt.xticks(rotation=45)
plt.show()

```



← **predicted dataset**, of **Jan1, 2025-Feb-1,2025** generated using available data up to **January 1, 2025**



← **actual dataset** of the market in **Jan1,2025 - Feb1,2024,** extracted from

www.nepsealpha.com

6. Comparison of Predicted and Actual Data Trends

❖ The line graph above represents the **predicted dataset**, generated using available data up to **January 1, 2025**. The predictions were tested against real market data from **January 1 to February 2025**, sourced from **NEPSE Alpha**.

A comparison of both graphs shows that the predicted data closely aligns with real-world market trends, demonstrating the model's **high accuracy and strong predictive capability**. However, minor discrepancies were observed in the dates. Upon deep analysis, we found that these errors occurred due to the **unexpected closure of the share market on public holidays**, such as **Lhosar**. Since our model did not account for such closures, it displayed market status on days when the market was actually closed, leading to slight misalignment in dates.

Despite this minor issue, the overall **nature of the curve perfectly aligns with actual market trends**, showcasing the model's **best performance** in forecasting stock movements. Future improvements, such as incorporating market closure data into the model, will further enhance its accuracy.

Results

The multivariate regression model was trained using historical stock data, considering ***Open*, *High*, *Low*, and *Volume*** as independent variables to predict the ***Closing Price***. After training and testing, the model's performance was evaluated using the following measures:

- **Mean Absolute Error (MAE):** Measures the average absolute difference between actual and predicted prices.
- **Mean Squared Error (MSE):** Indicates how much predictions deviate from actual values, with larger penalties for bigger errors.
- **R-squared (R^2):** Represents how well the independent variables explain the variation in stock prices. A higher R^2 suggests a better fit.

The model successfully identified patterns in stock price movements, achieving reasonable accuracy based on these measures.

Discussion

The multivariate regression model helped analyze how stock closing prices are affected by factors like the opening price, highest price, lowest price, and trading volume. The results showed that these factors have a strong influence on stock prices, making them useful for predictions. However, the model has some limitations because it does not consider outside influences like news, economic policies, or investor behavior, which can cause sudden changes in stock prices. It also assumes that the relationship between these factors and stock prices is always linear, which is not always true in real-world markets.The model was used to forecast future stock prices, and the predicted prices followed recent historical trends.However, real-world stock prices can be highly volatile, and additional features (e.g., interest rates, company earnings reports) could improve accuracy.Overall, this study shows that multivariate regression is a helpful tool for stock price forecasting but can be made better with further.

Conclusion

This project explored the use of ***multivariate regression analysis*** to predict stock prices based on historical data. By analyzing key factors such as ***opening price, high price, low price, and trading volume***, the model was able to capture trends and relationships influencing the closing price of NMB Bank shares. The evaluation measures showed that the model performed reasonably well, but it also had some limitations as it did not include ***external factors*** like market news and economic trends, which greatly influence stock prices.

Team:

Ankit Palanchoke	| Roll no: 29
Shubham Kayastha	| Roll no: 23
Sushan Sakha	| Roll no: 36
Swikar Shrestha	| Roll no: 00