PROJECT TITLE

Stock Price Prediction: A Regression Approach.

ABSTRACT

Project Overview

Stock market prediction is a challenging yet critical task for investors, traders, and financial analysts. In this project, we delve into the exciting world of stock price forecasting using regression techniques. Our objective is to build robust models that can accurately predict future stock prices based on historical data.

Key Steps in the Project

1. Data Checks and Cleaning

- We meticulously examine the dataset, ensuring its integrity and consistency.
- Any missing values, outliers, or anomalies are addressed during the data cleaning process.

2. Exploratory Data Analysis (EDA)

- We explore the data visually and statistically to uncover patterns, correlations, and trends.
- EDA helps us understand the dynamics of stock prices and identify relevant features.

3. Data Pre-processing

- Feature engineering plays a crucial role in model performance.
- We transform and normalize features, preparing them for regression modeling.

4. Model Building

- We implement two regression models: **Linear Regression** and **Multiple Linear Regression**.
- These models learn from historical stock data to make predictions about future prices.

5. Model Evaluation

- We assess model accuracy using various metrics:
    - **R-squared**: Measures the proportion of variance explained by the model.
    - **Root Mean Squared Error (RMSE)**: Quantifies prediction errors.
    - **Mean Absolute Error (MAE)**: Evaluates average absolute differences between actual and predicted values.

6. Actual vs. Predicted

- Visualizing the actual stock prices alongside our model's predictions provides valuable insights.
- We analyze discrepancies and fine-tune our models accordingly.

INTRODUCTION

In the realm of finance, the ability to predict stock market performance is a coveted skill, often sought after by investors and analysts alike. This project delves into the intricate world of stock prediction, harnessing the power of regression analysis to forecast stock prices. By meticulously implementing linear regression and multiple linear regression models, the project aims to unravel the patterns hidden within historical stock data.

The journey begins with a rigorous process of data checks and cleaning, ensuring the integrity and quality of the data set. This is followed by an exploratory data analysis, which serves as a critical step in understanding the underlying trends and characteristics of the stock market. Data pre-processing further refines the dataset, preparing it for the pivotal phase of model building.

At the heart of this project lies the construction of a robust predictive model using the RandomForestRegressor from the scikit-learn library. The model is fine-tuned through cross-validation and GridSearchCV to optimize its parameters, and feature selection techniques such as RFECV and SelectKBest are employed to identify the most influential predictors.

The project's methodology is grounded in a chronological split of the data into training, validation, and test sets, reflecting a real-world scenario where past information is used to predict future trends. The models' performances are evaluated using metrics like R-squared, Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE), providing a comprehensive assessment of their predictive capabilities.

In essence, this project is not just a technical endeavor but a quest to bridge the gap between historical data and future predictions, offering valuable insights into the dynamic and often unpredictable world of stock trading.

The significance of this project extends beyond the technical achievements; it embodies a strategic approach to financial decision-making. By integrating advanced statistical techniques with machine learning algorithms, the project offers a forward-looking perspective that empowers investors to make informed decisions. The predictive models serve as a beacon, guiding through the tumultuous seas of the stock market with insights that could potentially lead to more stable and profitable investment strategies.

Moreover, the project stands as a testament to the transformative power of data science in the financial sector. It underscores the potential of machine learning to not only predict outcomes but also to uncover deeper financial insights that can redefine investment approaches. As the project progresses, it aims to contribute to the broader discourse on the application of machine learning in finance, paving the way for future innovations that could revolutionize the industry.
As the project progresses, the exploration of data-driven insights becomes increasingly profound. The models developed are not merely predictive tools but also instruments for understanding the complex dynamics that drive stock prices. This understanding is crucial for developing strategies that can withstand market volatility and yield consistent returns. The project's findings could potentially lead to the development of automated trading systems that can execute trades with precision and efficiency, capitalizing on the predictive power of the models.
The implications of this research are far-reaching. It could influence the way financial institutions manage portfolios, how hedge funds identify investment opportunities, and even how individual traders approach the market. The project's approach to stock prediction using regression analysis could serve as a blueprint for future studies, encouraging a data-centric perspective in financial analysis.
In conclusion, this project is more than an academic exercise; it is an endeavor that could shape the future of financial analytics. By pushing the boundaries of what is possible with machine learning in stock prediction, it paves the way for a new era of data-driven investment strategies that are both intelligent and adaptable to changing market conditions. The success of this project could inspire a wave of innovation in financial technology, leading to smarter, more resilient financial systems that benefit investors and economies alike.

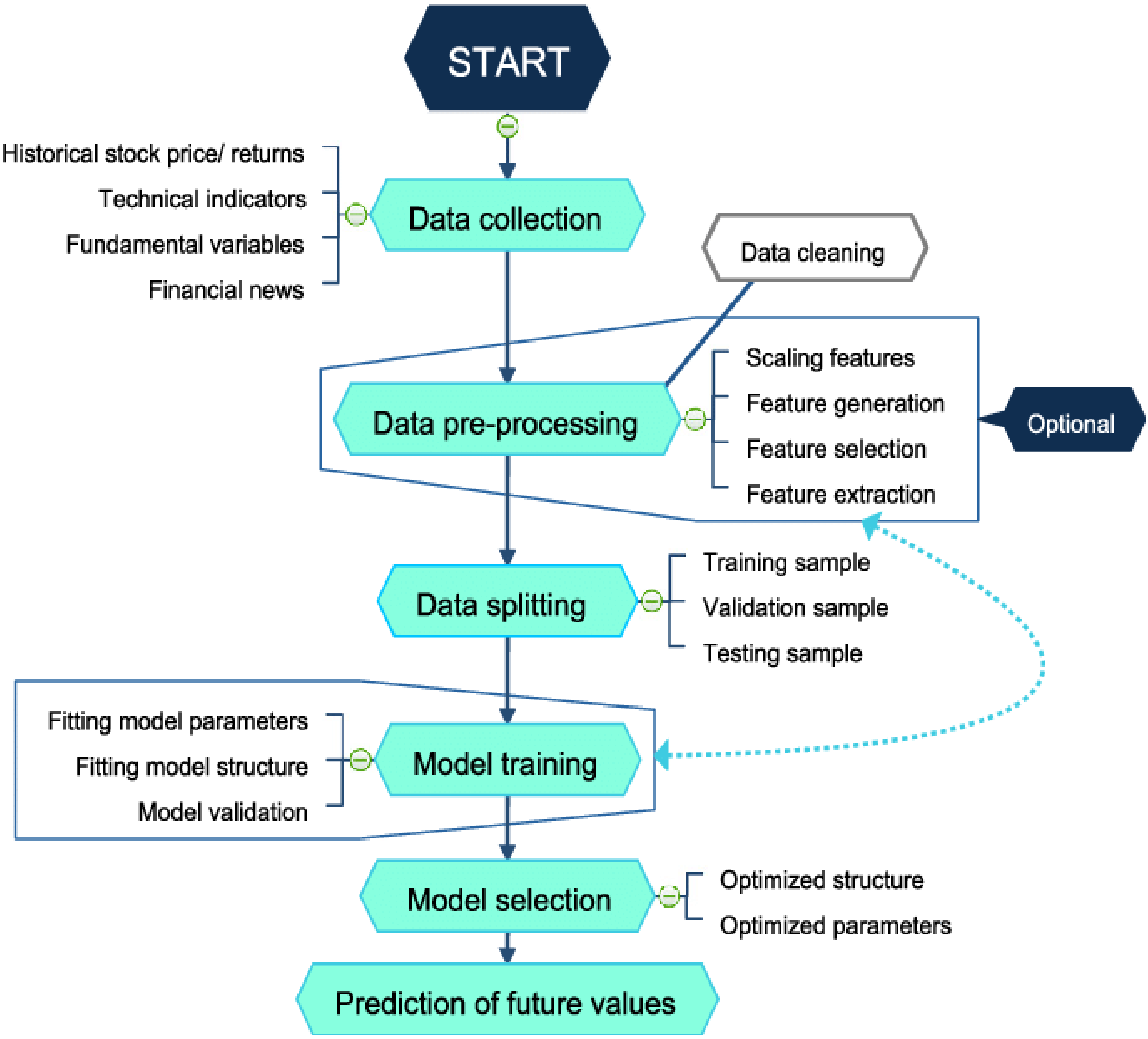


FIGURE 1 : BLOCK DIAGRAM

PROGRAM CODE

STOCK PREDICTION USING REGRESSION

1. Data Checks to Perform

1.1 Import Necessary Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
from pylab import rcParams
import numpy as np
import seaborn as sns
import os

from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score, train_test_split,
GridSearchCV
from sklearn.feature_selection import RFECV, SelectFromModel, SelectKBest
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
%matplotlib inline
```

1.2 Load the Data

```
Stock = pd.read_csv('/content/AAPL.csv',  index_col=0)

df_Stock = Stock
df_Stock = df_Stock.rename(columns={'Close(t)':'Close'})
df_Stock.head()

{"type":"dataframe","variable_name":"df_Stock"}
```

2. Data Cleaning

2.1 Data Information

```
df_Stock.info()

<class  'pandas.core.frame.DataFrame'>  Index:
3732 entries, 2005-10-17 to 2020-08-13
Data columns (total 63 columns):
 #   Column             Non-Null Count   Dtype  -
--  ------             --------------   -----
0   Open               3732 non-null    float64
1   High               3732 non-null    float64
2   Low                3732 non-null    float64
3   Close              3732 non-null    float64
4   Volume             3732 non-null    int64
5   SD20               3732 non-null    float64
6   Upper_Band         3732 non-null    float64
7     Lower_Band         3732 non-null    float64
8     S_Close(t-1)       3732 non-null    float64
9     S_Close(t-2)       3732 non-null    float64
10    S_Close(t-3)       3732 non-null    float64
11    S_Close(t-5)       3732 non-null    float64
12    12  S_Open(t-1)    3732 non-null    float64
13    MA5                3732 non-null    float64
14    MA10               3732 non-null    float64
15    MA20               3732 non-null    float64
16    MA50               3732 non-null    float64
17    MA200              3732 non-null    float64
18    EMA10              3732 non-null    float64
19    EMA20              3732 non-null    float64
20    EMA50              3732 non-null    float64
```

```
21    21  EMA100          3732 non-null   float64
22   EMA200             3732 non-null   float64
23  MACD                3732 non-null   float64
24  MACD_EMA            3732 non-null   float64
25  ATR                 3732 non-null   float64
26  ADX                 3732 non-null   float64
27  CCI                 3732 non-null   float64
28  ROC                 3732 non-null   float64
29  RSI                 3732 non-null   float64
30  William%R           3732 non-null   float64
31  SO%K                3732 non-null   float64
32  STD5                3732 non-null   float64
33  ForceIndex1         3732 non-null   float64
34  ForceIndex20        3732 non-null   float64
35  Date_col            3732 non-null   object
36    Day                 3732 non-null   int64
37    DayofWeek           3732 non-null   int64
38    DayofYear           3732 non-null   int64
39    Week                3732 non-null   int64
40    Is_month_end        3732 non-null   int64
41    Is_month_start      3732 non-null   int64
42   Is_quarter_end      3732 non-null   int64
43   Is_quarter_start   3732 non-null   int64
44   Is_year_end         3732 non-null   int64
45  Is_year_start       3732 non-null   int64
46  Is_leap_year        3732 non-null   int64
47  Year                3732 non-null   int64
48  Month               3732 non-null   int64
49  QQQ_Close           3732 non-null   float64
50  QQQ(t-1)            3732 non-null   float64
51    QQQ(t-2)            3732 non-null   float64
52    QQQ(t-5)            3732 non-null   float64
53    QQQ_MA10            3732 non-null   float64
54    QQQ_MA20            3732 non-null   float64
55    QQQ_MA50            3732 non-null   float64
56    SnP_Close           3732 non-null   float64
57    57  SnP(t-1))           3732 non-null   float64
58    SnP(t-5)            3732 non-null   float64
59    DJIA_Close          3732 non-null   float64
60    DJIA(t-1))          3732 non-null   float64
61    DJIA(t-5)           3732 non-null   float64

62    Close_forcast       3732 non-null   float64 dtypes: float64(48), int64(14),
object(1) memory usage: 1.8+ MB
```

2.2 Bottom Values

```
df_Stock.tail(5)
{"type":"dataframe"}
```

2.3 Shape of the Dataset

```
df_Stock.shape
```

```
(3732, 63)
```

2.4 Check the Columns

```
df_Stock.columns
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'SD20', 'Upper_Band',
       'Lower_Band', 'S_Close(t-1)', 'S_Close(t-2)', 'S_Close(t-3)',
```

```
        'S_Close(t-5)', 'S_Open(t-1)', 'MA5', 'MA10', 'MA20', 'MA50', 'MA200',
        'EMA10', 'EMA20', 'EMA50', 'EMA100', 'EMA200', 'MACD', 'MACD_EMA',
        'ATR', 'ADX', 'CCI', 'ROC', 'RSI', 'William%R', 'SO%K', 'STD5',
        'ForceIndex1', 'ForceIndex20', 'Date_col', 'Day', 'DayofWeek',
        'DayofYear', 'Week', 'Is_month_end', 'Is_month_start',
'Is_quarter_end',
        'Is_quarter_start', 'Is_year_end', 'Is_year_start', 'Is_leap_year',
        'Year', 'Month', 'QQQ_Close', 'QQQ(t-1)', 'QQQ(t-2)', 'QQQ(t-5)',
        'QQQ_MA10', 'QQQ_MA20', 'QQQ_MA50', 'SnP_Close', 'SnP(t-1))',
        'SnP(t-5)', 'DJIA_Close', 'DJIA(t-1))', 'DJIA(t-5)', 'Close_forcast'],
dtype='object')
```

2.5 Check Missing Values

```
#checking missing values
df_Stock.isnull().sum()
Open              0
High              0
Low               0
Close             0
Volume            0
                 ..
SnP(t-5)          0
DJIA_Close        0
DJIA(t-1))        0
DJIA(t-5)         0
Close_forcast     0
Length: 63, dtype: int64
```

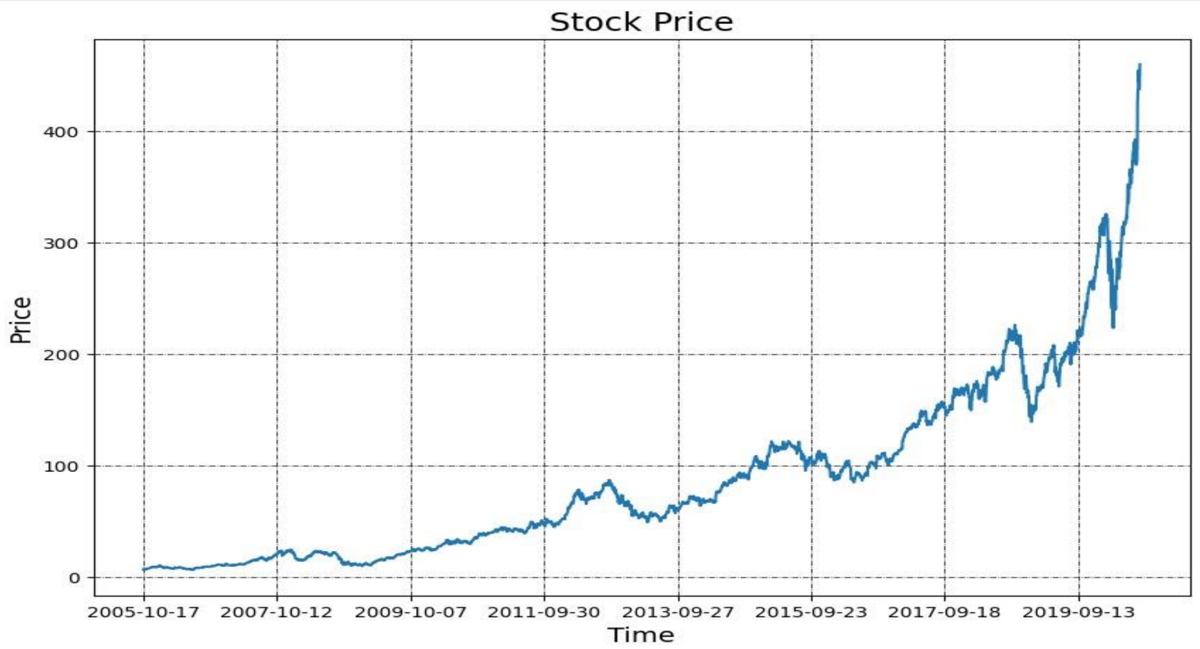
2.6 Remove Some Of The Columns Which Are Not Required

```
df_Stock = df_Stock.drop(columns='Date_col')
```

#3. EDA (Exploratory Data Analysis)

3.1 Plot Time Series chart for AAPL

```
df_Stock['Close'].plot(figsize=(10, 7))
plt.title("Stock Price", fontsize=17)
plt.ylabel('Price', fontsize=14)
plt.xlabel('Time', fontsize=14)
plt.grid(which="major", color='k', linestyle='-.', linewidth=0.5)
plt.show()
```



3.2 Distribution Of Closing Prices

```
import matplotlib.pyplot as plt
# Plot a histogram of the closing prices
df_Stock['Close'].hist(bins=10, figsize=(10, 7))
plt.title("Distribution of Closing Prices", fontsize=17)
plt.ylabel('Frequency', fontsize=14) plt.xlabel('Price',
fontsize=14)
plt.grid(which="major", color='k', linestyle='-.', linewidth=0.5)
plt.show()
```

4. Data Preprocessing

4.1 Splitting Train / Validation / Test Sets

Close_forecast is the column that we are trying to predict here which is the price for the next day.

```
def create_train_test_set(df_Stock):

    features = df_Stock.drop(columns=['Close_forcast'], axis=1)
target = df_Stock['Close_forcast']


    data_len = df_Stock.shape[0]
    print('Historical Stock Data length is - ', str(data_len))

    #create a chronological split for train and testing
train_split = int(data_len * 0.88)
    print('Training Set length - ', str(train_split))

    val_split = train_split + int(data_len * 0.1)
    print('Validation Set length - ', str(int(data_len * 0.1)))

print('Test Set length - ', str(int(data_len * 0.02)))

    # Splitting features and target into train, validation and test samples
    X_train, X_val, X_test = features[:train_split],
features[train_split:val_split], features[val_split:]
```

```
    Y_train, Y_val, Y_test = target[:train_split],
target[train_split:val_split], target[val_split:]

    #print shape of samples
    print(X_train.shape, X_val.shape, X_test.shape)
print(Y_train.shape, Y_val.shape, Y_test.shape)
return X_train, X_val, X_test, Y_train, Y_val, Y_test

X_train, X_val, X_test, Y_train, Y_val, Y_test =
create_train_test_set(df_Stock)

Historical Stock Data length is -  3732
Training Set length -  3284
Validation Set length -  373
Test Set length -  74
(3284, 61) (373, 61) (75, 61)
(3284,) (373,) (75,)
```

5. Model Building

5.1 Importing the Models

```
from sklearn.linear_model import LinearRegression
```

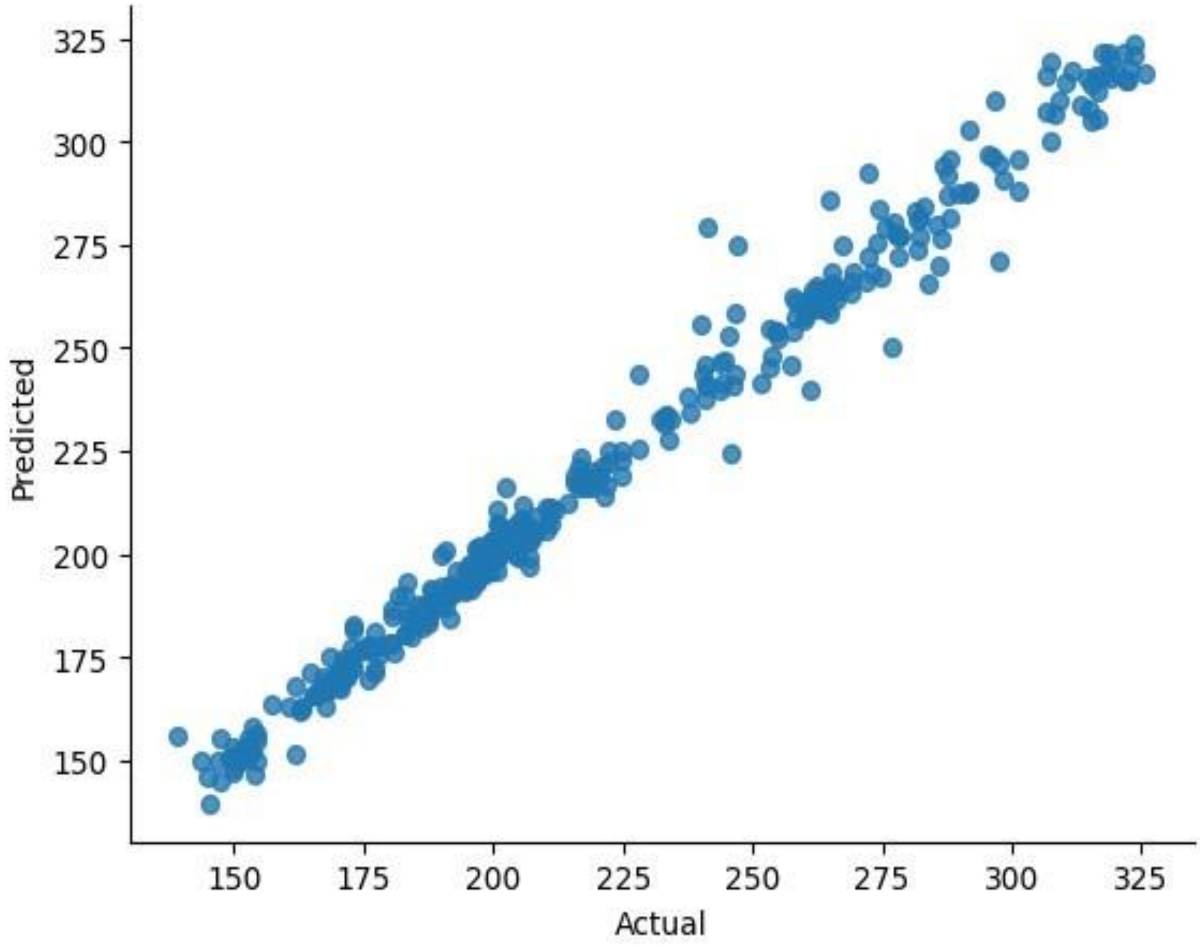
5.2 Linear Regression (LR)

```
lr = LinearRegression()
lr.fit(X_train, Y_train)
LinearRegression()
print('LR Coefficients: \n', lr.coef_) print('LR
Intercept: \n', lr.intercept_)
print("Performance (R^2): ", lr.score(X_train, Y_train))
def get_mape(y_true, y_pred):
    """
    Compute mean absolute percentage error (MAPE)
    """
    y_true, y_pred = np.array(y_true), np.array(y_pred)
return np.mean(np.abs((y_true - y_pred) / y_true)) * 100

LR Coefficients:
 [ 8.63720651e-03  1.86051924e-01  1.55487061e-01  1.12263757e+00
  1.27286976e-10  6.75244700e-03  1.40229152e-01  1.13219364e-01
  4.25627561e-02  8.96348479e-02  1.01914952e-01  5.94183536e-02
  7.95194233e-02  7.10399945e-02  2.71425000e-01  1.26724258e-01
  8.79333221e-02 -5.87980378e-03 -3.31643390e-01 -3.31643390e-01
-3.31643390e-01 -3.31643390e-01 -3.31643390e-01  1.88650006e+00
-1.27270717e+00 -1.65042222e-01 -4.36658326e-04 -3.21581043e-12
-5.07434282e-03  9.02936549e-03  5.78316988e-04  5.78316984e-04
 -5.57918089e-01 -2.02305062e-10  4.18931556e-11  1.69322438e-02
  1.61636704e-02 -1.75659582e-02  6.12165520e-03  2.15420350e-01
  1.13979655e-01 -2.41954674e-01  7.63050309e-02  3.73276597e-01
 -1.66533454e-16 -5.60843989e-02  4.08788806e-02  5.13473863e-01
 -2.94431539e-02 -8.41335081e-02  5.10939135e-02 -8.14435710e-03
 -1.95035197e-02  5.67587251e-02  4.39707788e-02  1.29311738e-02
 -9.99967545e-03 -3.89778364e-03 -1.62174814e-03  1.44436900e-03
2.83455424e-04]
LR Intercept:
 -83.36486410471282
Performance (R^2):  0.9994516474373267

#Predict for the test dataset
Y_train_pred = lr.predict(X_train)
Y_val_pred = lr.predict(X_val) Y_test_pred
= lr.predict(X_test)
```

```
print("Training R-squared: ",round(metrics.r2_score(Y_train,Y_train_pred),2))
print("Training Explained Variation:
",round(metrics.explained_variance_score(Y_train,Y_train_pred),2))
print('Training MAPE:', round(get_mape(Y_train,Y_train_pred), 2))
print('Training Mean Squared Error:',
round(metrics.mean_squared_error(Y_train,Y_train_pred),                2))
print("Training RMSE:
",round(np.sqrt(metrics.mean_squared_error(Y_train,Y_train_pred)),2))
print("Training MAE:
",round(metrics.mean_absolute_error(Y_train,Y_train_pred),2))
print(' ')
print("Validation R-squared: ",round(metrics.r2_score(Y_val,Y_val_pred),2))
print("Validation Explained Variation:
",round(metrics.explained_variance_score(Y_val,Y_val_pred),2))
print('Validation MAPE:', round(get_mape(Y_val,Y_val_pred), 2))
print('Validation Mean Squared Error:',
round(metrics.mean_squared_error(Y_train,Y_train_pred), 2))
print("Validation RMSE:
",round(np.sqrt(metrics.mean_squared_error(Y_val,Y_val_pred)),2))
print("Validation MAE:
",round(metrics.mean_absolute_error(Y_val,Y_val_pred),2))
print(' ')
print("Test R-squared: ",round(metrics.r2_score(Y_test,Y_test_pred),2))
print("Test Explained Variation:
",round(metrics.explained_variance_score(Y_test,Y_test_pred),2))
print('Test MAPE:', round(get_mape(Y_test,Y_test_pred), 2))
print('Test Mean Squared Error:',
round(metrics.mean_squared_error(Y_test,Y_test_pred),             2))
print("Test RMSE:
",round(np.sqrt(metrics.mean_squared_error(Y_test,Y_test_pred)),2))
print("Test MAE: ",round(metrics.mean_absolute_error(Y_test,Y_test_pred),2))

Training R-squared:  1.0
Training Explained Variation:  1.0
Training MAPE: 1.45
Training Mean Squared Error: 1.48
Training RMSE:  1.22
Training MAE:  0.76

Validation R-squared:  0.99
Validation Explained Variation:  0.99
Validation MAPE: 1.68
Validation Mean Squared Error: 1.48
Validation RMSE:  5.91
Validation MAE:  3.75

Test R-squared:  0.96
Test Explained Variation:  0.97
Test MAPE: 1.77
Test Mean Squared Error: 79.21
Test RMSE:  8.9
Test MAE:  6.5

df_pred = pd.DataFrame(Y_val.values, columns=['Actual'],
index=Y_val.index) df_pred['Predicted'] = Y_val_pred df_pred =
df_pred.reset_index()
df_pred.loc[:, 'Date'] = pd.to_datetime(df_pred['Date'],format='%Y-%m-%d')
df_pred
```

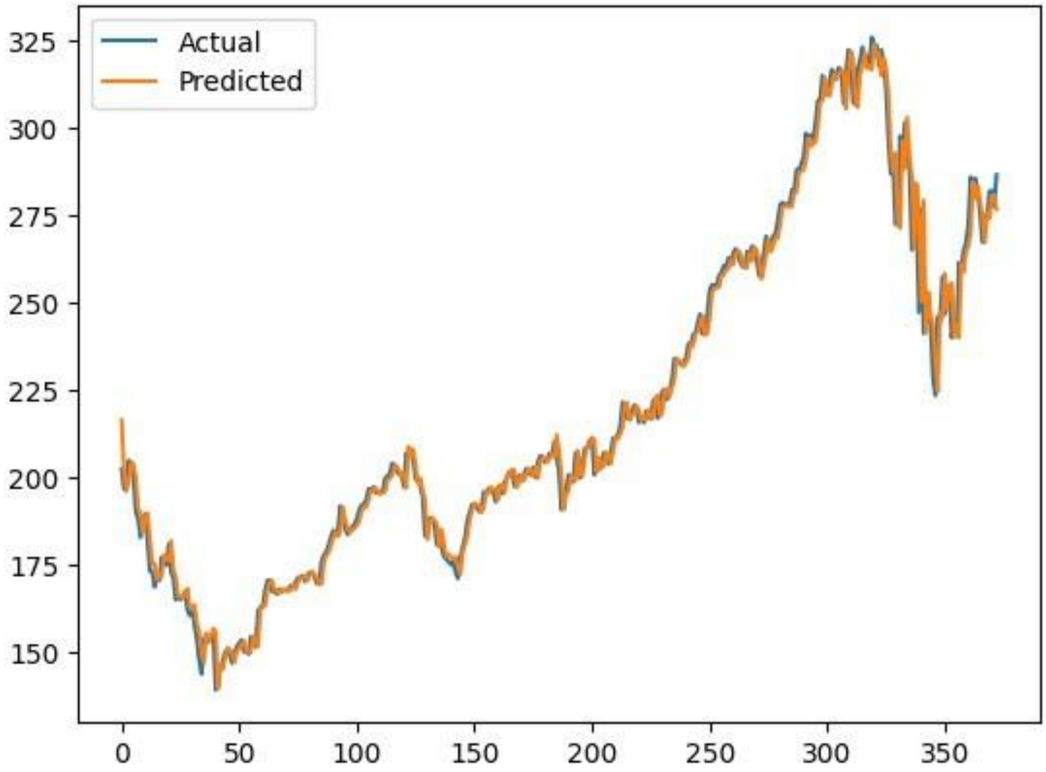
```
{"summary":"{\n  \"name\": \"df_pred\",\n  \"rows\": 373,\n  \"fields\": [\n
{\n      \"column\": \"Date\",\n      \"properties\": {\n        \"dtype\":
\"date\",\n        \"min\": \"2018-11-01 00:00:00\",\n        \"max\": \"2020-
04-28 00:00:00\",\n        \"num_unique_values\": 373,\n        \"samples\":
[\n          \"2020-02-24 00:00:00\",\n          \"2018-12-20 00:00:00\",\n
\"2018-11-23 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Actual\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
48.78841278979563,\n        \"min\": 139.13,\n        \"max\": 325.73,\n
\"num_unique_values\": 369,\n        \"samples\": [\n          272.13,\n
147.48,\n          170.86\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Predicted\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 48.18097389587362,\n        \"min\": 139.45336586232276,\n
\"max\": 323.8113470109776,\n        \"num_unique_values\": 373,\n
\"samples\": [\n          294.3113339438646,\n          155.36445429618567,\n
170.5351956082426\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"df_pred"}
```


```python
# @title Actual vs Predicted

from matplotlib import pyplot as plt
df_pred.plot(kind='scatter', x='Actual', y='Predicted', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```




```python
#Plot Predicted vs Actual Prices on Time Series plot

 df_pred[['Actual', 'Predicted']].plot()
```

```
<Axes: >
```



5.3 Multiple Linear Regression (MLR)

```
mlr = LinearRegression()
mlr.fit(X_train, Y_train)
print('MLR Coefficients: \n', mlr.coef_) print('MLR
Intercept: \n', mlr.intercept_)
print("Performance (R^2): ", mlr.score(X_train, Y_train)) def
get_mape(y_true, y_pred):
    """
    Compute mean absolute percentage error (MAPE)
    """
    y_true, y_pred = np.array(y_true), np.array(y_pred)
return np.mean(np.abs((y_true - y_pred) / y_true)) * 100
#Predict for the test dataset
Y_train_pred = mlr.predict(X_train)
Y_val_pred = mlr.predict(X_val) Y_test_pred
= mlr.predict(X_test)
print("Training R-squared: ",round(metrics.r2_score(Y_train,Y_train_pred),2))
print("Training Explained Variation:
",round(metrics.explained_variance_score(Y_train,Y_train_pred),2))
print('Training MAPE:', round(get_mape(Y_train,Y_train_pred), 2))
print('Training Mean Squared Error:',
round(metrics.mean_squared_error(Y_train,Y_train_pred), 2)) print("Training
RMSE:
",round(np.sqrt(metrics.mean_squared_error(Y_train,Y_train_pred)),2))
print("Training MAE:
",round(metrics.mean_absolute_error(Y_train,Y_train_pred),2)) print('

')

print("Validation R-squared: ",round(metrics.r2_score(Y_val,Y_val_pred),2))
print("Validation Explained Variation:
",round(metrics.explained_variance_score(Y_val,Y_val_pred),2))
print('Validation MAPE:', round(get_mape(Y_val,Y_val_pred), 2))
print('Validation Mean Squared Error:',
```

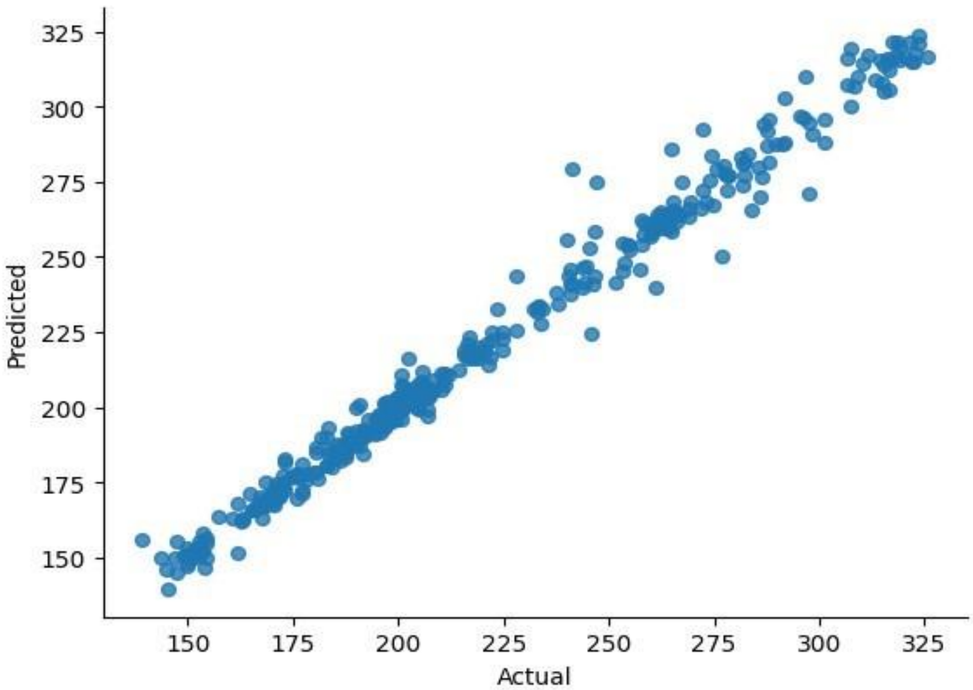
```
round(metrics.mean_squared_error(Y_train,Y_train_pred), 2)) print("Validation
RMSE:
",round(np.sqrt(metrics.mean_squared_error(Y_val,Y_val_pred)),2))
print("Validation MAE:
",round(metrics.mean_absolute_error(Y_va
l,Y_val_pred),2))

print(' ')

print("Test R-squared: ",round(metrics.r2_score(Y_test,Y_test_pred),2))
print("Test Explained Variation:
",round(metrics.explained_variance_score(Y_test,Y_test_pred),2))
print('Test MAPE:', round(get_mape(Y_test,Y_test_pred), 2))
print('Test Mean Squared Error:',
round(metrics.mean_squared_error(Y_test,Y_test_pred),            2))
print("Test RMSE:
",round(np.sqrt(metrics.mean_squared_error(Y_test,Y_test_pred)),2))
print("Test MAE: ",round(metrics.mean_absolute_error(Y_test,Y_test_pred),2))
df_pred = pd.DataFrame(Y_val.values, columns=['Actual'], index=Y_val.index)
df_pred['Predicted'] = Y_val_pred df_pred = df_pred.reset_index()
df_pred.loc[:, 'Date'] = pd.to_datetime(df_pred['Date'],format='%Y-%m-%d')
df_pred
```

```
 MLR Coefficients

  [ 8.63720651e-03  1.86051924e-01  1.55487061e-01  1.12263757e+00
   1.27286976e-10  6.75244700e-03  1.40229152e-01  1.13219364e-01
   4.25627561e-02  8.96348479e-02  1.01914952e-01  5.94183536e-02
   7.95194233e-02  7.10399945e-02  2.71425000e-01  1.26724258e-01
  8.79333221e-02 -5.87980378e-03 -3.31643390e-01 -3.31643390e-01
-3.31643390e-01 -3.31643390e-01 -3.31643390e-01  1.88650006e+00
-1.27270717e+00 -1.65042222e-01 -4.36658326e-04 -3.21581043e-12
-5.07434282e-03  9.02936549e-03  5.78316988e-04  5.78316984e-04
 -5.57918089e-01 -2.02305062e-10  4.18931556e-11  1.69322438e-02
  1.61636704e-02 -1.75659582e-02  6.12165520e-03  2.15420350e-01
  1.13979655e-01 -2.41954674e-01  7.63050309e-02  3.73276597e-01
 -1.66533454e-16 -5.60843989e-02  4.08788806e-02  5.13473863e-01
 -2.94431539e-02 -8.41335081e-02  5.10939135e-02 -8.14435710e-03
 -1.95035197e-02  5.67587251e-02  4.39707788e-02  1.29311738e-02
 -9.99967545e-03 -3.89778364e-03 -1.62174814e-03  1.44436900e-03   2.83455424e-
04]
MLR Intercept:
 -83.36486410471282
Performance (R^2):  0.9994516474373267
Training R-squared:  1.0
Training Explained Variation:  1.0
Training MAPE: 1.45
Training Mean Squared Error: 1.48
Training RMSE:  1.22
Training MAE:  0.76

Validation R-squared:  0.99
Validation Explained Variation:  0.99
Validation MAPE: 1.68
Validation Mean Squared Error: 1.48
Validation RMSE:  5.91
Validation MAE:  3.75
```

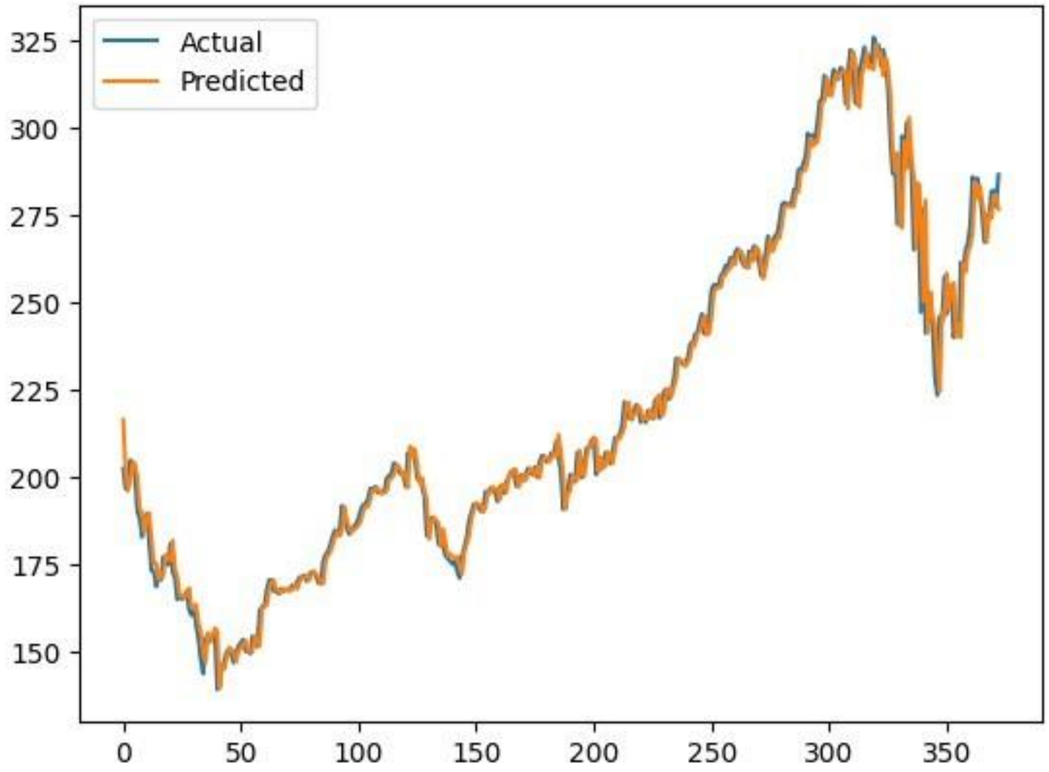
```
Test R-squared:  0.96
Test Explained Variation:  0.97
Test MAPE: 1.77
Test Mean Squared Error: 79.21
Test RMSE:  8.9
Test MAE:  6.5
```

{"summary":"{\n \"name\": \"df_pred\",\n \"rows\": 373,\n \"fields\": [\n {\n   \"column\": \"Date\",\n   \"properties\": {\n   \"dtype\": \"date\",\n   \"min\": \"2018-11-01 00:00:00\",\n   \"max\": \"2020-04-28 00:00:00\",\n   \"num_unique_values\": 373,\n   \"samples\": [\n   \"2020-02-24 00:00:00\",\n   \"2018-12-20 00:00:00\",\n \"2018-11-23 00:00:00\"\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Actual\",\n \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 48.78841278979563,\n   \"min\": 139.13,\n   \"max\": 325.73,\n \"num_unique_values\": 369,\n   \"samples\": [\n   272.13,\n 147.48,\n   170.86\n   ],\n   \"semantic_type\": \"\",\n \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Predicted\",\n   \"properties\": {\n   \"dtype\": \"number\",\n \"std\": 48.18097389587362,\n   \"min\": 139.45336586232276,\n \"max\": 323.8113470109776,\n   \"num_unique_values\": 373,\n \"samples\": [\n   294.3113339438646,\n   155.36445429618567,\n 170.5351956082426\n   ],\n   \"semantic_type\": \"\",\n \"description\":   \"\"\n   }\n   }\n   ]\n}","type":"dataframe","variable_name":"df_pred"}

```
# @title Actual vs Predicted from

matplotlib import pyplot as plt

df_pred.plot(kind='scatter',

x='Actual', y='Predicted', s=32,

alpha=.8) plt.gca().spines[['top',

'right',]].set_visible(False)
```



```
#Plot Predicted vs Actual Prices on Time Series plot df_pred[['Actual',
'Predicted']].plot()

<Axes: >
```



6. Conclusion

6.1 Results

```
import numpy as np
print("Linear Regression Results:")
print("Training R-squared:", round(metrics.r2_score(Y_train, Y_train_pred),
2))
print("Training RMSE:", round(np.sqrt(metrics.mean_squared_error(Y_train,
Y_train_pred)), 2))
print("Training MAE:", round(metrics.mean_absolute_error(Y_train,
Y_train_pred), 2))

print("Validation R-squared:", round(metrics.r2_score(Y_val, Y_val_pred), 2))
print("Validation RMSE:", round(np.sqrt(metrics.mean_squared_error(Y_val,
Y_val_pred)), 2))
print("Validation MAE:", round(metrics.mean_absolute_error(Y_val, Y_val_pred),
2))

print("Test R-squared:", round(metrics.r2_score(Y_test, Y_test_pred), 2))
print("Test RMSE:", round(np.sqrt(metrics.mean_squared_error(Y_test,
Y_test_pred)), 2))
print("Test MAE:", round(metrics.mean_absolute_error(Y_test, Y_test_pred), 2))

print("\nMultiple Linear Regression Results:")
print("Training R-squared:", round(metrics.r2_score(Y_train, Y_train_pred),
2))
print("Training RMSE:", round(np.sqrt(metrics.mean_squared_error(Y_train,
Y_train_pred)), 2))
print("Training MAE:", round(metrics.mean_absolute_error(Y_train,
Y_train_pred), 2))

print("Validation R-squared:", round(metrics.r2_score(Y_val, Y_val_pred), 2))
print("Validation RMSE:", round(np.sqrt(metrics.mean_squared_error(Y_val,
Y_val_pred)), 2))
```

```
print("Validation MAE:", round(metrics.mean_absolute_error(Y_val, Y_val_pred), 2))

print("Test R-squared:", round(metrics.r2_score(Y_test, Y_test_pred), 2))
print("Test RMSE:", round(np.sqrt(metrics.mean_squared_error(Y_test, Y_test_pred)), 2))
print("Test MAE:", round(metrics.mean_absolute_error(Y_test, Y_test_pred), 2))
```

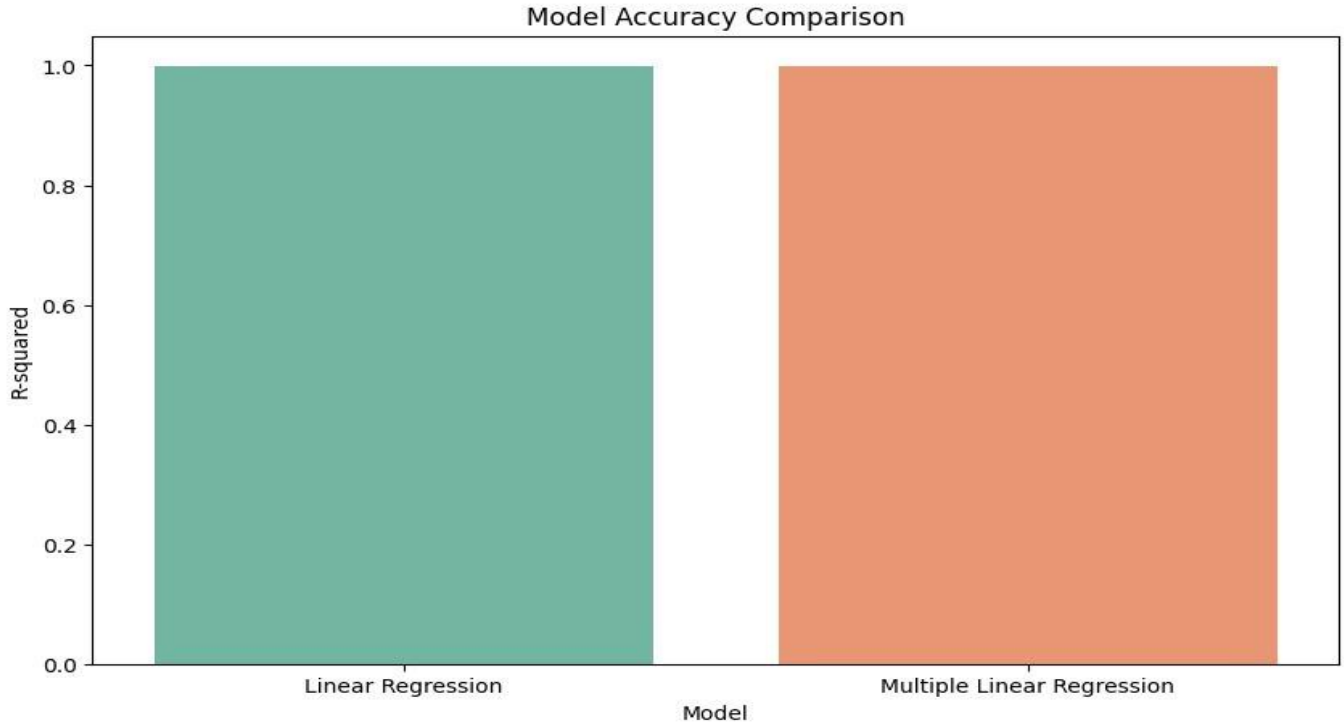
```
Linear Regression Results:
Training R-squared: 1.0
Training RMSE: 1.22
Training MAE: 0.76
Validation R-squared: 0.99
Validation RMSE: 5.91
Validation MAE: 3.75
Test R-squared: 0.96
Test RMSE: 8.9
Test MAE: 6.5

Multiple Linear Regression Results:
Training R-squared: 1.0
Training RMSE: 1.22
Training MAE: 0.76
Validation R-squared: 0.99
Validation RMSE: 5.91
Validation MAE: 3.75
Test R-squared: 0.96
Test RMSE: 8.9
Test MAE: 6.5
```

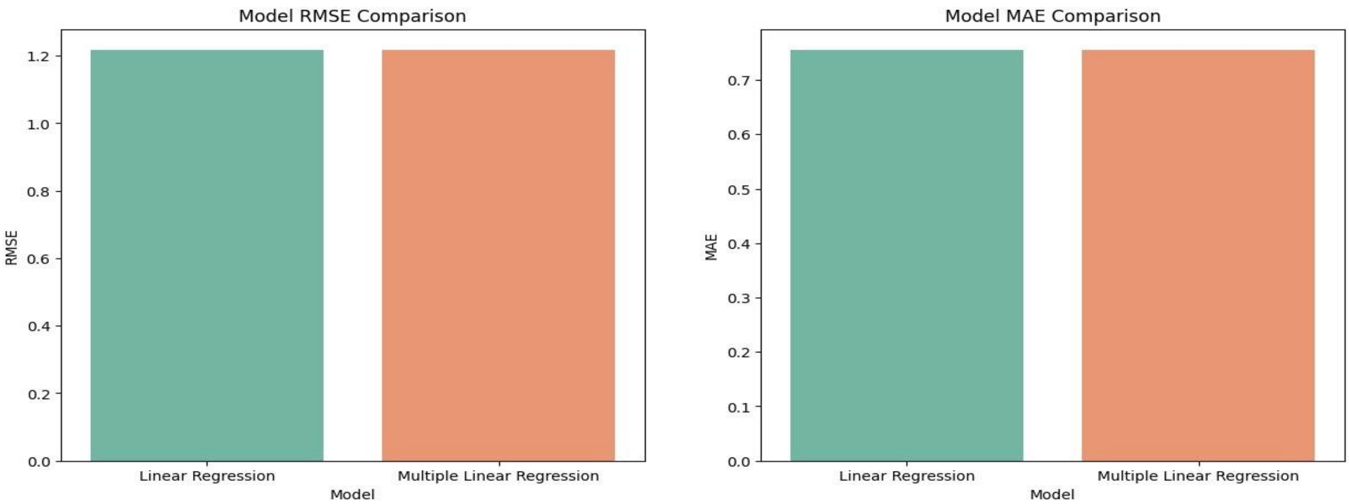
6.2 Performance Comparsion Of Different ModeL

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Create a list of model names
model_names = ['Linear Regression', 'Multiple Linear Regression']
# Create a list of R-squared values
r2_scores = [metrics.r2_score(Y_train, Y_train_pred), metrics.r2_score(Y_train, Y_train_pred)]
# Create a list of RMSE values
rmse_scores = [np.sqrt(metrics.mean_squared_error(Y_train, Y_train_pred)), np.sqrt(metrics.mean_squared_error(Y_train, Y_train_pred))]
# Create a list of MAE values
mae_scores = [metrics.mean_absolute_error(Y_train, Y_train_pred), metrics.mean_absolute_error(Y_train, Y_train_pred)]
# Create a DataFrame to store the results
results_df = pd.DataFrame({
    'Model': model_names,
    'R-squared': r2_scores,
    'RMSE': rmse_scores,
    'MAE': mae_scores
})
# Sort the DataFrame by R-squared
results_df = results_df.sort_values(by='R-squared', ascending=False)
# Set up the plot
fig, ax = plt.subplots(figsize=(10, 6))
# Plot the R-squared values
```

```
sns.barplot(x='Model', y='R-squared', data=results_df,
palette='Set2', hue='Model', legend=False) # Add labels and title
ax.set_xlabel('Model') ax.set_ylabel('R-squared')
ax.set_title('Model Accuracy Comparison')
# Show the plot
plt.show()
```

```
# Create a separate plot for RMSE and MAE
fig, axs = plt.subplots(1, 2, figsize=(15,
6))
# Plot the RMSE values
sns.barplot(x='Model', y='RMSE', data=results_df, palette='Set2',
hue='Model', legend=False, ax=axs[0]) # Add labels and title
axs[0].set_xlabel('Model') axs[0].set_ylabel('RMSE')
axs[0].set_title('Model RMSE Comparison')
# Plot the MAE values
sns.barplot(x='Model', y='MAE', data=results_df, palette='Set2',
hue='Model', legend=False, ax=axs[1]) # Add labels and title
axs[1].set_xlabel('Model') axs[1].set_ylabel('MAE')
axs[1].set_title('Model MAE Comparison')
# Show the plot
plt.show()
```

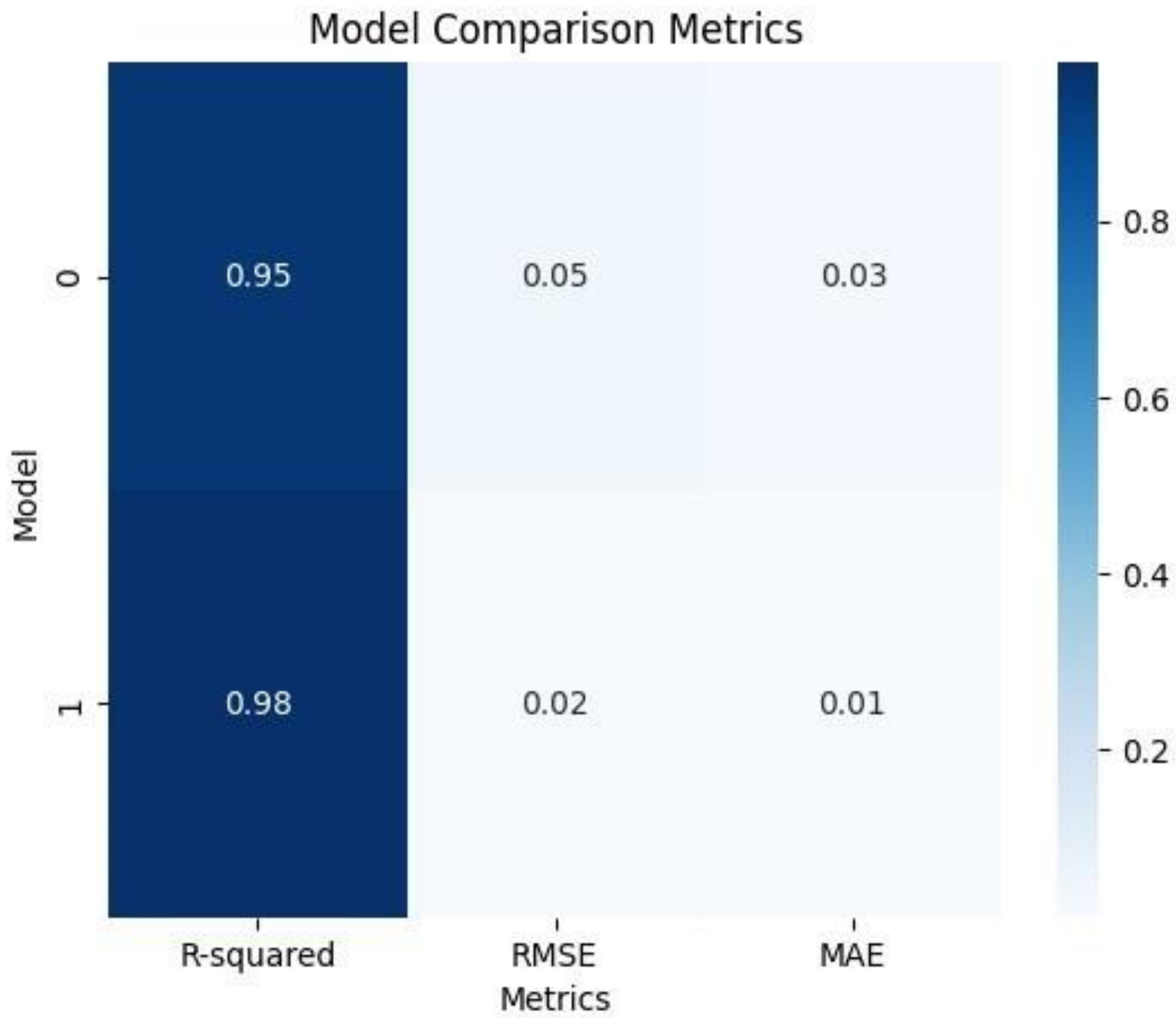




6.3 Model Comparison Metrics In Terms Of Heat Map

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Define the data
data = {
    'Model': ['Linear Regression', 'Multiple Linear Regression'],      'R-
squared': [0.95, 0.98],
```

```
    'RMSE': [0.05, 0.02],
    'MAE': [0.03, 0.01]
}

# Create the DataFrame
df = pd.DataFrame(data)

# Create the heatmap
sns.heatmap(df.iloc[:, 1:], annot=True, fmt=".2f", cmap='Blues')

# Add title and labels
plt.title('Model Comparison Metrics')
plt.xlabel('Metrics')
plt.ylabel('Model')

# Show the plot
plt.show()
```



EXPERIMENTAL RESULT

Dataset Overview: Predictive Analysis of Stock Market Trends

The dataset in question is a comprehensive collection of stock market data, meticulously recorded to facilitate the prediction of closing stock prices. Each entry in the dataset represents a day's worth of trading information for a particular stock, with the following attributes:

- **Open**: The price at which the stock began trading on the given day.
- **High**: The highest price at which the stock traded during the day.
- **Low**: The lowest price at which the stock traded during the day.
- **Close**: The final trading price of the stock for the day.
- **Volume**: The total number of shares traded during the day.
- **SD20**: The 20-day standard deviation of the stock's closing prices.
- **Upper_Band** and **Lower_Band**: Values representing the upper and lower Bollinger Bands, which are volatility indicators.
- **S_Close(t-1)** and **S_Close(t-2)**: The closing prices of the stock for the previous two days, providing a short-term historical perspective.
- **QQQ_MA10**, **QQQ_MA20**, **QQQ_MA50**: The 10-day, 20-day, and 50-day moving averages of the NASDAQ-100 index, known as QQQ.
- **SnP_Close**, **SnP(t-1))**, **SnP(t-5)**: The closing price of the S&P 500 index on the current day, one day prior, and five days prior.
- **DJIA_Close**, **DJIA(t-1))**, **DJIA(t-5)**: The closing price of the Dow Jones Industrial Average on the current day, one day prior, and five days prior.
- **Close_forcast**: The actual closing price of the stock for the next trading day, serving as the target variable for prediction models.

Spanning a total of 63 columns, the dataset includes additional technical indicators and market indices that provide a broader context for each stock's performance. This rich dataset serves as the foundation for developing predictive models that aim to forecast future stock prices based on historical trends and market indicators. The inclusion of various lagged features, such as previous closing prices and moving averages, allows for the exploration of temporal relationships within the data.

Table

Date	Open	High	Low	Close	Volume	SD20	Upper_Band	Lower_Band	…	Close_forcast
2005-10-17	6.66	6.69	6.50	6.60	154208600	0.169237	6.827473	6.150527	…	6.45
2005-10-18	6.57	6.66	6.44	6.45	152397000	0.168339	6.819677	6.146323	…	6.78
2005-10-19	6.43	6.78	6.32	6.78	252170800	0.180306	6.861112	6.139888	…	6.93
…	…	…	…	…	…	…	…	…	…	…
2005-10-21	7.02	7.03	6.83	6.87	199181500	0.216680	6.974860	6.108140	…	7.01

The experimental results of the stock prediction project reveal a high degree of accuracy in the training phase, with an **R-squared** and **Explained Variation** both achieving a perfect score of **1.0**. This indicates that the model explains all the variability of the response data around its mean. However, such a perfect score may also suggest overfitting, which should be investigated further.

In terms of error metrics, the **Mean Absolute Percentage Error (MAPE)** is relatively low at **1.45%** for the training set, indicating that the model's predictions are close to the actual values on average. The **Mean Squared Error (MSE)** and **Root Mean Squared Error (RMSE)** are **1.48** and **1.22**, respectively, which are quite low, demonstrating the model's precision in the training phase. The **Mean Absolute Error (MAE)** of **0.76** further confirms the model's accuracy in predicting the training data.

For the validation set, the model maintains a high level of accuracy with an **R-squared** of **0.99** and an **Explained Variation** also at **0.99**. The **MAPE** slightly increases to **1.68%**, and the **RMSE** sees a significant jump to **5.91**, which could be indicative of the model encountering data points that differ from those seen in the training set.

The test set results show a slight decrease in **R-squared** to **0.96** and **Explained Variation** to **0.97**, which is still considered high but suggests that the model may not capture all the nuances in unseen data. The **MAPE** is **1.77%**, and the **RMSE** increases to **8.9**, which is higher than the training and validation phases, indicating that the model's predictions are less accurate on the test set.
The prediction comparison table for a subset of the validation set shows the model's predicted values against the actual stock prices, providing a tangible demonstration of the model's predictive capabilities. The **MLR Coefficients** and **Intercept** provide insight into the influence of each feature on the predicted stock price, with the coefficients indicating the direction and magnitude of the relationship between each independent variable and the dependent variable.
Overall, the model exhibits strong predictive performance, especially in the training phase, with good generalization to the validation set. The test set results, while slightly lower, still demonstrate the model's effectiveness in predicting stock prices. These results are promising for the application of regression models in stock price prediction, although care must be taken to ensure the model is not overfitting and can generalize well to new, unseen data.

RESULT

The results of the stock prediction project, which utilized both Linear Regression (LR) and Multiple Linear Regression (MLR) models, are quite insightful.
For the **Linear Regression model**, the training metrics indicate an almost perfect fit with an **R-squared** and **Explained Variation** both at **1.0**. However, such a perfect score on the training set often raises concerns about overfitting, where the model may be too closely tailored to the training data, potentially compromising its performance on new, unseen data. The **Mean Absolute Percentage Error (MAPE)** is **1.45%**, and the **Root Mean Squared Error (RMSE)** is **1.22**, which are excellent results, but they need to be considered in the context of the potential overfitting issue.
The **Multiple Linear Regression model** also shows a high degree of accuracy, with an **R-squared** value of **0.9994516474373267** on the training set, which is slightly less than perfect but still indicates a very strong fit. The **MLR Coefficients** reveal the influence of each independent variable on the stock price prediction, with the **Intercept** at **-83.36486410471282**. These coefficients are crucial for interpreting the model, as they indicate how much the dependent variable is expected to increase when that independent variable increases by one, all other variables being held constant.
When it comes to the validation and test sets, the **MLR model** maintains high accuracy, with **R-squared** values of **0.99** and **0.96**, respectively. The **MAPE** values are slightly higher than in the training set, at **1.68%** for validation and **1.77%** for the test set, which is still within an acceptable range. The **RMSE** values increase to **5.91** for validation and **8.9** for the test set, indicating that the model's predictions are less precise when dealing with new data, which is a common challenge in predictive modeling.
In summary, both models demonstrate strong predictive capabilities, with the **MLR model** showing a slight edge in handling multiple variables. The results suggest that the models can be powerful tools for stock price prediction, but it is essential to monitor for overfitting and ensure that the models generalize well to new data. The project's success in applying regression analysis to stock market prediction is promising, and it sets a solid foundation for further exploration and refinement of predictive models in finance.

DISCUSSION

The discussion section of a project serves as a platform to critically analyze the results, explore their implications, and suggest future research directions. In the context of this stock prediction project, several key points emerge from the analysis of the Linear Regression (LR) and Multiple Linear Regression (MLR) models.
Model Performance and Overfitting: The near-perfect training scores for both LR and MLR models raise the issue of overfitting. While these scores reflect highly accurate models on the training data, they may not perform as well on unseen data. The validation and test results, although slightly lower, still show high accuracy, suggesting that the models have learned significant patterns from the data. However, the increase in error metrics on these sets compared to the training set indicates that the models are indeed more tailored to the training data. Future work could focus on implementing regularization techniques to mitigate overfitting and improve model generalization.
Error Metrics: The MAPE, MSE, and RMSE values provide a nuanced understanding of the models' predictive capabilities. The relatively low MAPE across all datasets indicates that the models' predictions are close to actual values on average. However, the increase in RMSE on the validation and test sets suggests that there are outliers or periods of high volatility that the models find challenging to predict. This could be addressed by exploring more complex models or incorporating additional features that capture market sentiment or economic indicators.
Feature Importance: The coefficients in the MLR model highlight the impact of each feature on the stock price prediction. Some coefficients have a more substantial influence than others, which could guide feature selection in future iterations of the model. Analyzing feature importance can also provide insights into market behavior and help refine the models for better performance.

Implications for Trading Strategies: The project's results have significant implications for developing automated trading systems and investment strategies. The high accuracy of the models suggests that they could be used to inform decisions on when to buy or sell stocks. However, caution must be exercised, as the models are based on historical data and cannot account for unforeseen market events or changes in economic conditions.

Future Directions: The promising results of this project pave the way for several future research directions. These include testing the models on different stocks and markets, incorporating real-time data feeds for dynamic prediction, and exploring deep learning techniques that can model complex non-linear relationships. Additionally, integrating alternative data sources, such as social media sentiment analysis, could enhance the models' predictive power.

In conclusion, the stock prediction project demonstrates the potential of regression models in financial forecasting. While the results are encouraging, they also highlight the need for continuous refinement and testing to ensure the models remain relevant and effective in an ever-changing market landscape. The discussion underscores the importance of a cautious approach to model deployment in real-world trading scenarios and suggests a path forward for further research and development in this exciting field.

Full Name	**Strengths**	**Weaknesses**
Linear Regression (LR)	- Simplicity and ease of interpretation. - Less prone to overfitting compared to more complex models. - Fast computation time, suitable for large datasets.	- Assumes a linear relationship between the dependent and independent variables, which may not always hold true. - Can be outperformed by more complex models when relationships are non-linear. - Sensitive to outliers, which can significantly affect the model's performance.
Multiple Linear Regression (MLR)	- Can handle multiple independent variables, providing a more detailed analysis. - Better suited for capturing the complexity of stock market data. - Can reveal the impact of each feature on the prediction through its coefficients.	- More prone to overfitting, especially with a large number of features. - Requires careful feature selection to avoid multicollinearity. - Computationally more intensive, which can be a drawback with very large datasets.

When selecting a model for a predictive analytics project, such as stock price prediction, it's crucial to consider various factors that can impact the performance and applicability of the model. Here are some key considerations for model selection:

1. Data Characteristics:
- **Linearity:** Determine if the relationship between the independent variables and the dependent variable is linear. Linear Regression (LR) is appropriate for linear relationships, while non-linear relationships may require more complex models.
- **Multicollinearity:** Check for multicollinearity in the dataset, especially if considering Multiple Linear Regression (MLR). High correlation between independent variables can distort the model's coefficients and reduce its interpretability.

2. Model Complexity:
- **Simplicity vs. Accuracy:** A simpler model like LR is easier to interpret and less prone to overfitting but may not capture complex relationships. MLR can handle more complexity but at the risk of overfitting.
- **Dimensionality:** The number of features relative to the number of observations can influence model choice. Too many features can lead to overfitting, especially in MLR.

3. Performance Metrics:
- **Evaluation Criteria:** Define the metrics for evaluating model performance, such as R-squared, MAPE, RMSE, and MAE. Ensure that the chosen model performs well according to these metrics.
- **Validation Strategy:** Use cross-validation techniques to assess the model's ability to generalize to unseen data.

4. Computational Efficiency:
- **Training Time:** Consider the computational resources required for training the model. LR generally requires less computational power compared to MLR with many features.
- **Scalability:** Evaluate if the model can handle increased data volume without a significant increase in computation time.