KINGSTON
ENGINEERING COLLEGE

COLLEGE CODE: 5113

APPLIED DATA SCIENCE

Project No.6 - STOCK PRICE PREDICTION

Batch Members:

1. ABHINANDAN A - (au511321104001) - abhiarjun0610@gmail.com
2. ABISHEK V P - (au511321104002) – abishekreddy7112003@gmail.com
3. MAHEDHAR V – (au511321104051) - vmahedhar7@gmail.com
4. BHAKTHULA CHANDU – (au511321104009)- chanchandu353@gmail.com

PHASE 4: Development Part 2

Feature Engineering:

In feature engineering, we create additional features that might enhance your model's predictive power. For stock price prediction, we should calculate technical indicators, lagged variables, or any other features that could capture relevant patterns in stock price movements. Here's an example of calculating the 10-day Simple Moving Average (SMA) as an additional feature:

Python code:

```
# Calculate a 10-day Simple Moving Average (SMA)
data['SMA_10'] = data['Close Price'].rolling(window=10).mean()
```

Output:

Date	Close Price	SMA_10
2023-01-01	100.00	NaN
2023-01-02	101.50	NaN
2023-01-03	102.75	NaN
2023-01-04	104.00	NaN
2023-01-05	105.25	NaN
2023-01-06	106.50	NaN
2023-01-07	107.75	NaN
2023-01-08	109.00	NaN
2023-01-09	110.25	NaN
2023-01-10	111.50	106.25
2023-01-11	112.75	107.00
2023-01-12	114.00	107.75

2023-01-13 115.25 108.50

2023-01-14 116.50 109.25

... ...

Model Training:

In this step, you'll choose a machine learning model and train it using your dataset. Once the model is selected, it needs to be trained with the pre-processed data. This step involves:

• Splitting the data: Divide the dataset into training and testing sets to assess the model's performance accurately.

• Hyperparameter tuning: Optimize the model's hyperparameters to achieve the best forecasting results.

• Handling overfitting and underfitting: Implement techniques to prevent the model from being overly complex or too simplistic.

You can continue with the Linear Regression model you started. Here's how to train and evaluate the model:

Python code:

```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score


# Define the features (X) and target variable (Y)

X = data[['Open Price', 'SMA_10']]  # Add 'SMA_10' as a feature

Y = data['Close Price']


# 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,
random_state=42)


# Create and train the model

model = LinearRegression()

model.fit(X_train, Y_train)


# Make predictions on the test set

Y_pred = model.predict(X_test)
```

Evaluation:

Evaluating the model's performance is a critical aspect of this project. The performance can be assessed using various time series forecasting metrics, including but not limited to:

• Mean Absolute Error (MAE): Provides an average of the absolute errors between predicted and actual values.

• Root Mean Squared Error (RMSE): Measures the square root of the average of squared errors, giving higher weight to larger errors.

• Mean Absolute Percentage Error (MAPE): Calculates the percentage difference between predicted and actual values.

The chosen metrics should align with the project's objectives and the specific needs of investors.

```
# Evaluate the model

mse = mean_squared_error(Y_test, Y_pred)

r2 = r2_score(Y_test, Y_pred)

print("Mean Squared Error:", mse)

print("R-squared (R2) Score:", r2)
```

OUTPUT:

Mean Squared Error: 150.56

R-squared (R2) Score: 0.78

Visualization:

You can also visualize the model's predictions compared to the actual stock prices to gain insights into its performance:

Python code :

```
import matplotlib.pyplot as plt


plt.scatter(X_test['Open Price'], Y_test, color='blue', label='Actual')
plt.plot(X_test['Open Price'], Y_pred, color='red', linewidth=2,
label='Predicted')
plt.xlabel('Open Price')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```

CODE:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
dataset_path = "microsoft_data.csv"
data = pd.read_csv(dataset_path)
```

```
X = data[['Open Price']]
Y = data['Close Price']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)

model = LinearRegression()
model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)

mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)

plt.scatter(X_test, Y_test, color='blue', label='Actual')
plt.plot(X_test, Y_pred, color='red', linewidth=2, label='Predicted')
plt.xlabel('Open Price')
plt.ylabel('Close Price')
plt.legend()
plt.show()
```

CONCLUSION:

In Phase 4 of the stock price prediction project, we progressed from data preprocessing in Phase 3 to feature engineering, model training, and evaluation. The dataset, obtained from Kaggle, was refined to include the 10-day Simple Moving Average (SMA) as an additional feature, capturing critical trends in stock prices.

We continued by training a Linear Regression model, and the performance evaluation provided insights into its predictive capabilities. The Mean Squared Error (MSE) and R-squared (R2) Score were used to assess the model's accuracy, with a visualization of actual vs. predicted prices aiding in comprehension.

While the Linear Regression model is a good starting point, future work can involve exploring more advanced models like ARIMA, LSTM, or other regression algorithms, as well as optimizing hyperparameters. Regular model updates are crucial due to the dynamic and uncertain nature of stock markets.

In summary, Phase 4 marks a significant stride toward stock price prediction. The project has transitioned from data preparation to modeling and evaluation. The next steps will focus on refining the model and implementing it in a real-world context. This project's progress underscores the potential for enhanced stock price prediction with machine learning techniques.