STOCK PRICE PREDICTION USING APPLIED DATA SCIENCE

Phase 4 submission document

Project Title: Stock Price Prediction

Phase 4: Development Part 2

submitted by :

YAALIN.S (911721106040)
BAVANI.E(911721106304)
JANAKI.V(911721106307)
KIRUBHASINI.I(911721106013)

Content:

In this part, I build the stock price prediction model by

- Feature engineering
- Model training
- Evaluation.

STOCK PRICE PREDICTION

INTRODUCTION:

- In the dynamic world of financial markets, the ability to forecast stock prices is a perennial challenge and a pursuit that has captivated investors and analysts for decades. Stock price prediction involves leveraging various methodologies, including traditional financial analysis and cutting- edge data science techniques, to anticipate the future movements of individual stocks or broader market indices.

- Stock price prediction involves using various methodologies, tools, and models to estimate the future movements of a stock's value in financial markets. Investors, traders, and financial analysts engage in stock price prediction to make informed decisions regarding buying, selling, or holding stocks.

- Feature selection is the process of identifying and selecting the mostrelevant features from a dataset to improve the performance of a machine learning model. This is an important step in building a stock price prediction model, as it can help to reduce overfitting and improve the generalization ability of the model.

- Model training is the process of feeding the selected features to a machine learning algorithm and allowing it to learn the relationshipbetween the features and the target variable (i.e., stock price). Once the model is trained, it can be used to predict the house prices of new houses given their features.

- Model evaluation is the process of assessing the performance of a trained machine learning model on a held-out test set. This is important to ensure that the model is generalizing well and that itis not overfitting the training data.

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 different features 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.

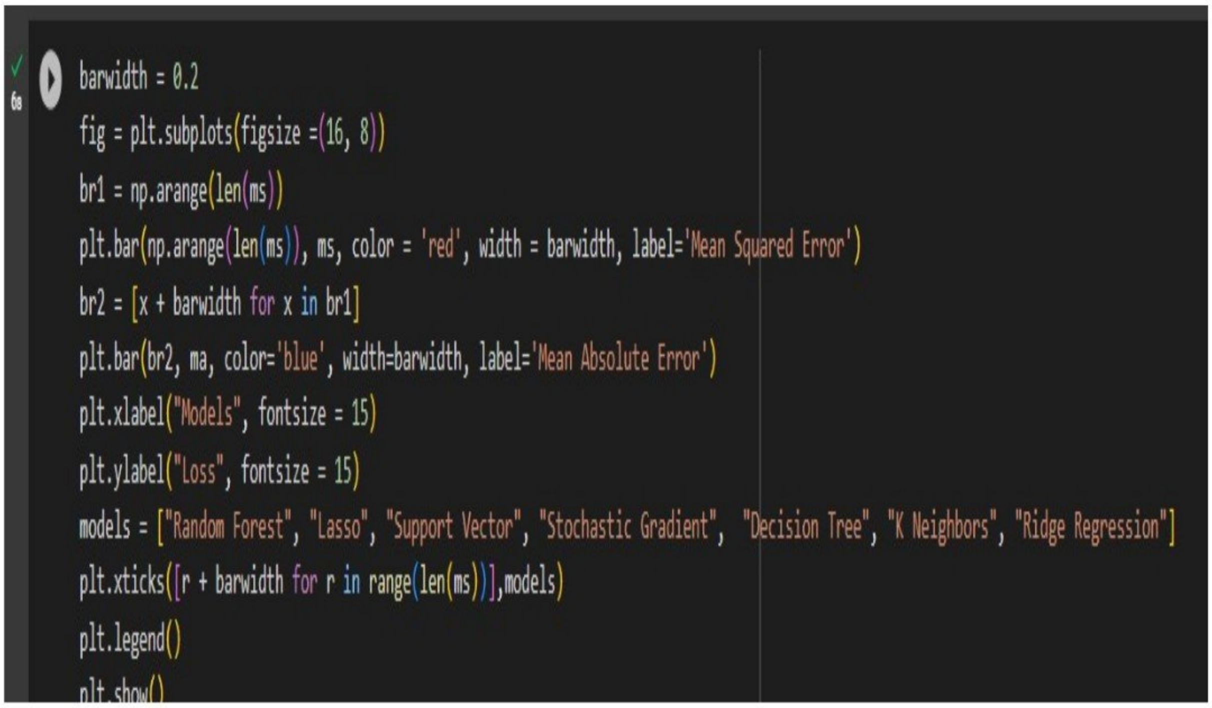
Model Training

- 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.

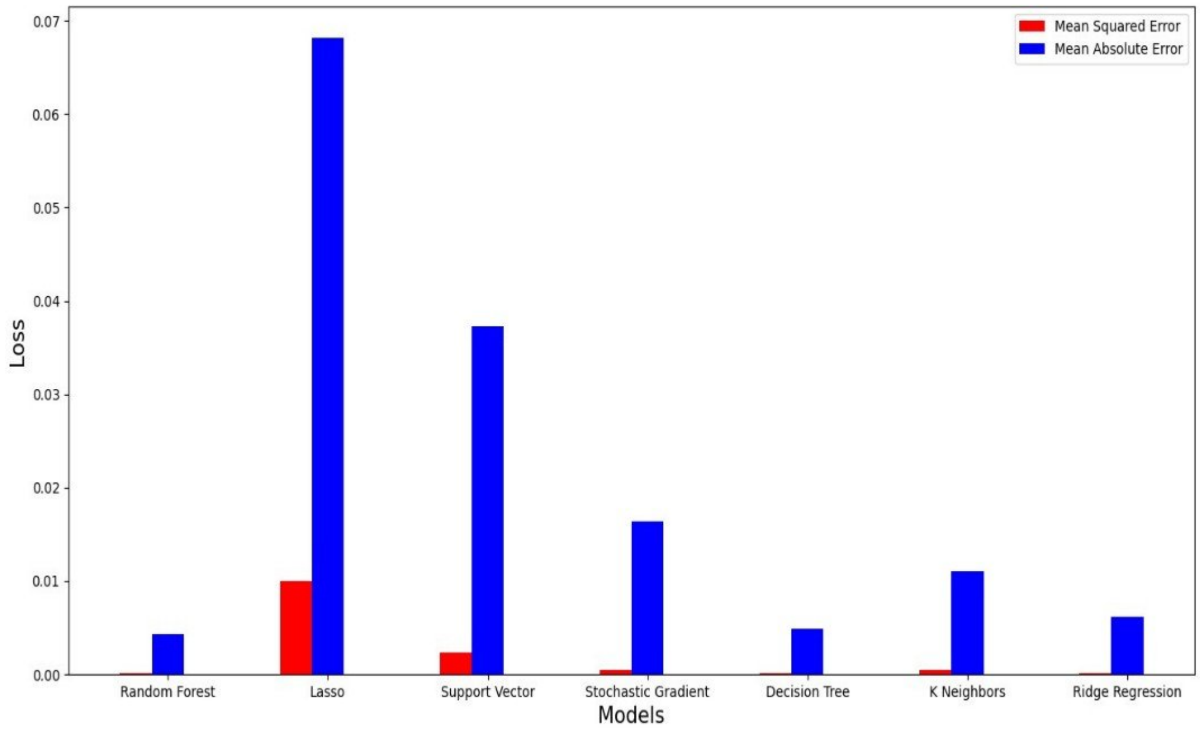
Model evaluation:

- 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.

Plotting Losses of different models

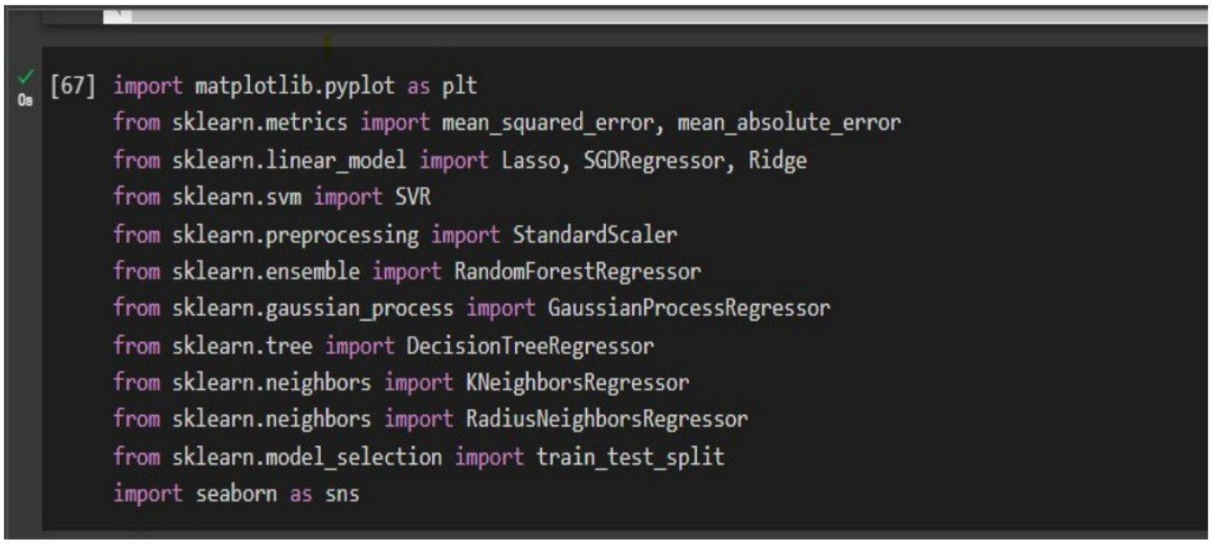


```
barwidth = 0.2
fig = plt.subplots(figsize =(16, 8))
br1 = np.arange(len(ms))
plt.bar(np.arange(len(ms)), ms, color = 'red', width = barwidth, label='Mean Squared Error')
br2 = [x + barwidth for x in br1]
plt.bar(br2, ma, color='blue', width=barwidth, label='Mean Absolute Error')
plt.xlabel("Models", 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()
```



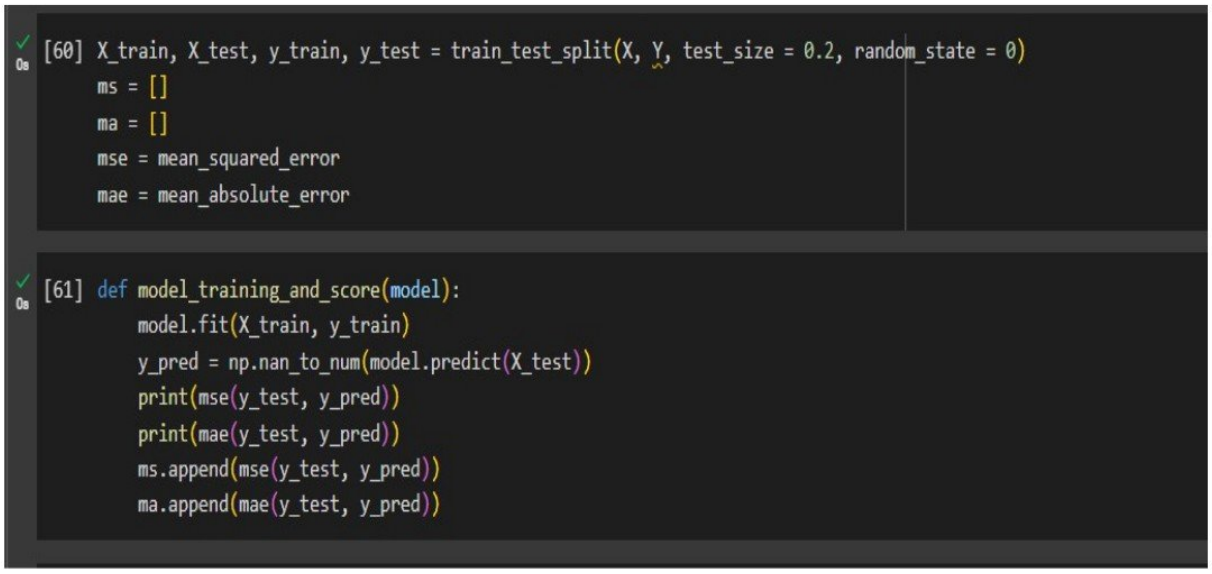
Some Common Techniques :

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
```

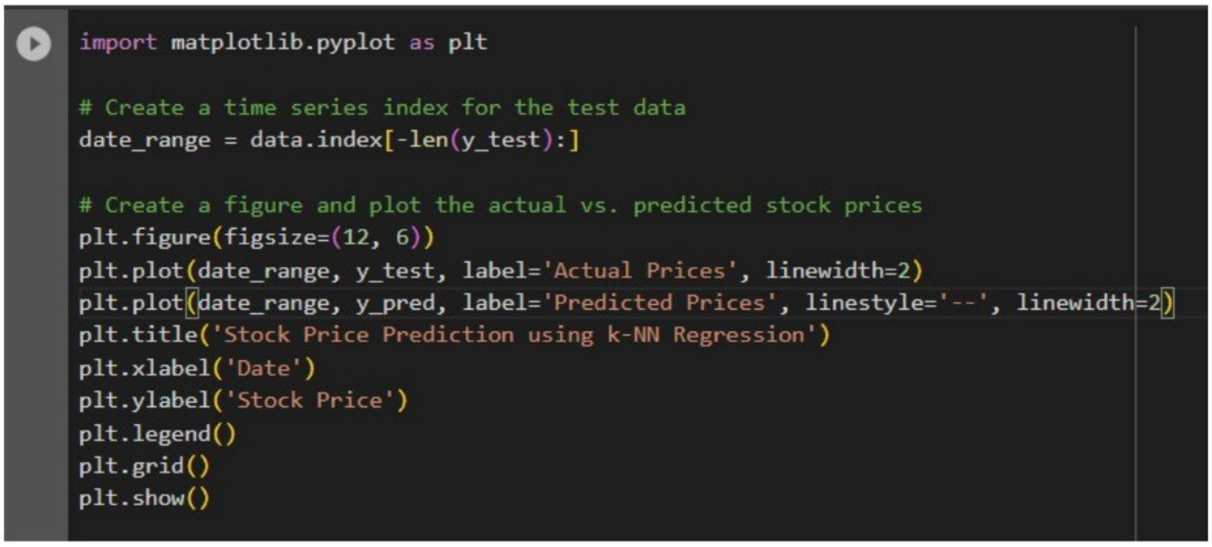
Model Training



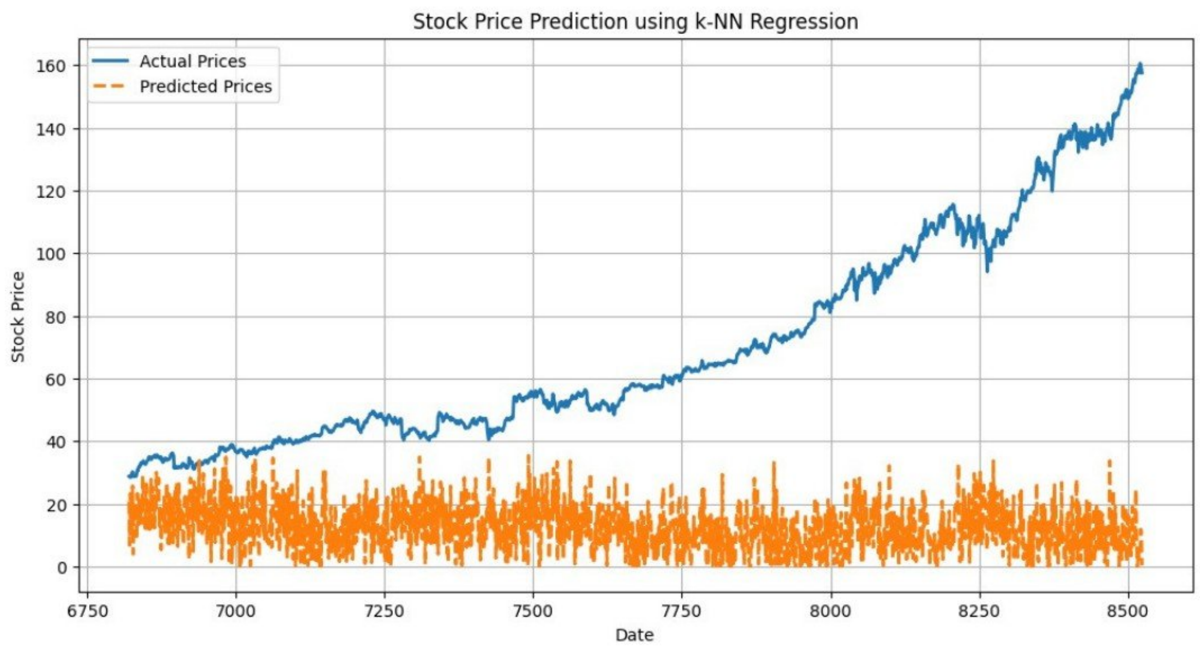
```
[60] X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
     ms = []
     ma = []
     mse = mean_squared_error
     mae = mean_absolute_error
```

```
[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))
```

Visualization of KNN

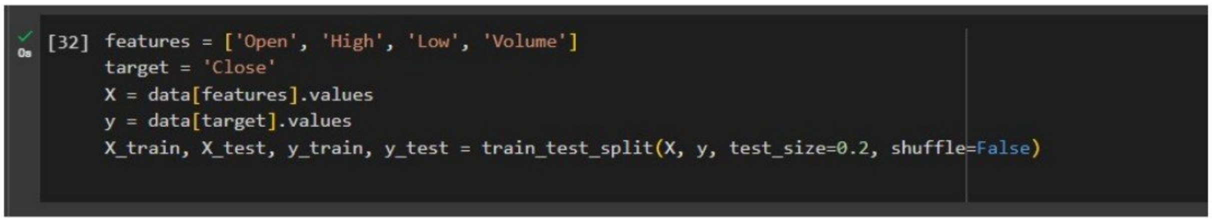


```
import matplotlib.pyplot as plt

# 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()
```



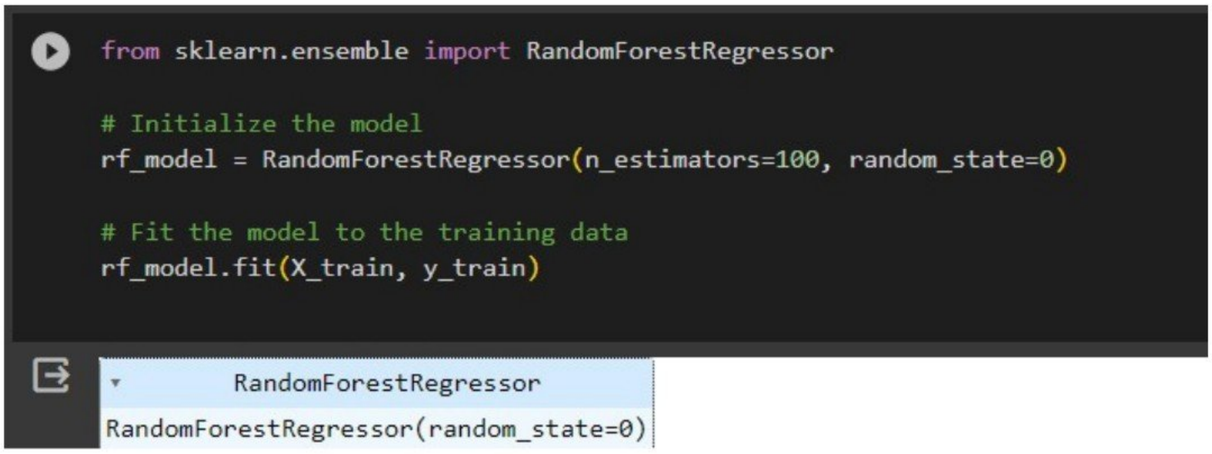
4. Random Forest

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)
```

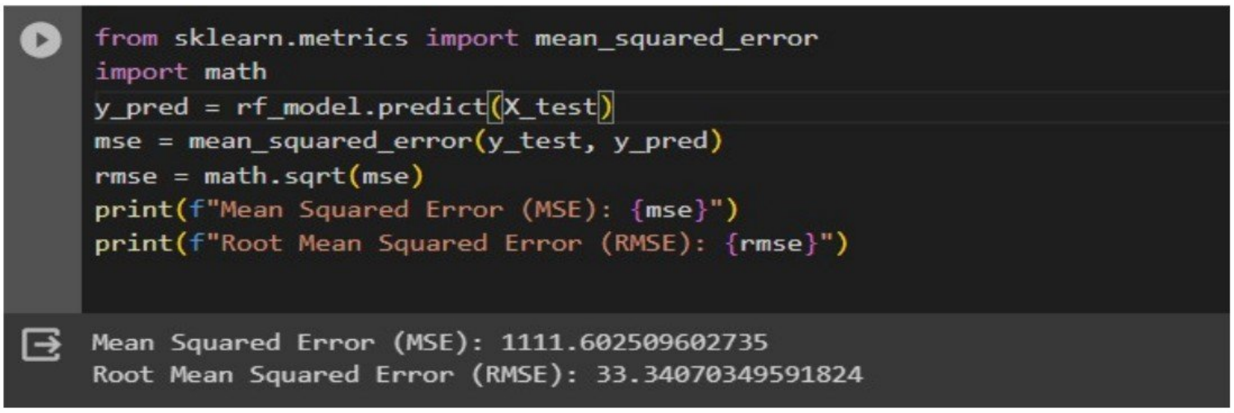
Model Training



```
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

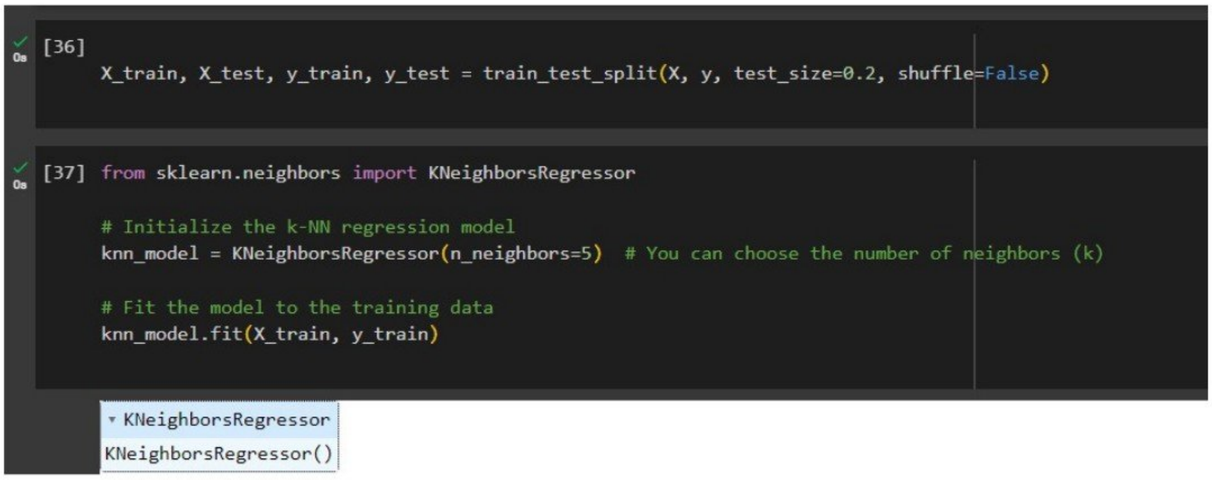


```
from sklearn.metrics import mean_squared_error
import math
y_pred = rf_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = math.sqrt(mse)
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
```

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

5. KNN

Model Training

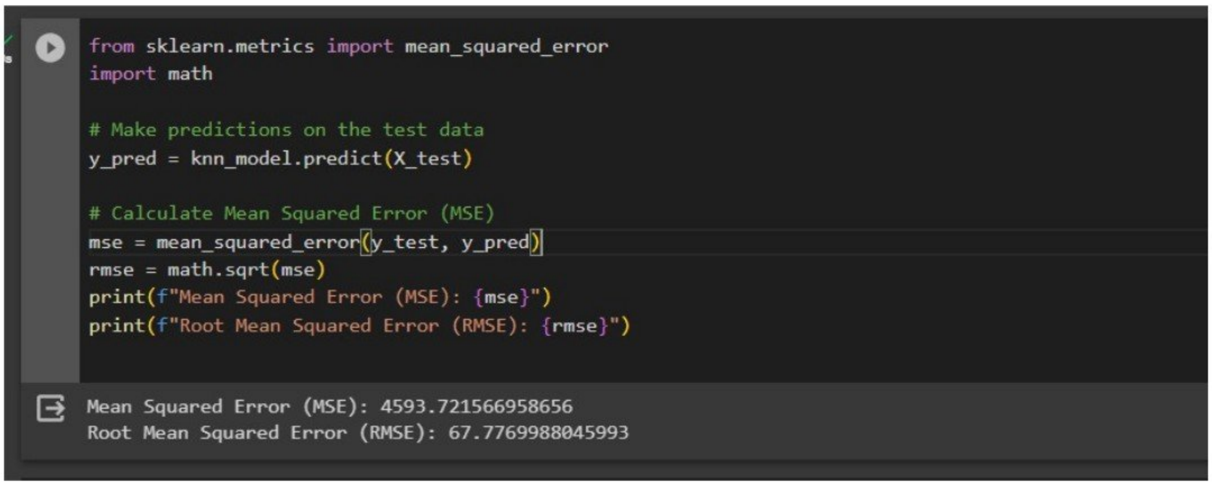


```
[36]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)
```

```
[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
```

CONCLUSION:

□

In conclusion, while data science techniques can provide valuable insights into stock price movements, the inherent complexity and uncertainty 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.