STOCK PRICE PREDICTION

In this part you will continue building your project. Continue building the stock price prediction model by

Feature engineering

Model training

Evaluation.

INTRODUCTION:

Predicting stock price is a challenging yet critical task for investors, traders and financial analysts. It involves harnessing the power of data science and machine learning. There are three essential components of stock price prediction: Feature Engineering and Model Training and Evaluation.

FEATURE ENGINEERING:

This step involves crafting meaningful input data for our model. we will extract and create relevant features, such as historical stock prices, trading volumes, and technical indicator. Here is the list of common features you might consider:

1.Historical Prices:

Daily or minute-level historical prices, including open, High, Low, Close(OHLC), and adjusted Close.

Code:

```
# importing the data set
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

```
import pandas as pd
df = pd.read_csv('MSFT.csv')
df
```

OUTPUT:

Prices

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

8525 rows × 7 columns

```
# OHLC prices
import matplotlib.pyplot as plt
plt.figure(figsize=(16,8))
plt.title('Close Price History', fontsize=18)
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

OUTPUT:

filtering the OHLC Prices

```
import math

import numpy as np

close_data = df.filter(['Close'])

dataset = close_data.values

training_data_len = math.ceil(len(dataset) * .8)


training_data_len
```

Output:

```
6820
```

```
# define the target value:
import pandas as pd
df = pd.read_csv('MSFT.csv')
data['Target'] = data['Close'].shift(-1)
#Drop missing values and the last row (since there's no target value)
import pandas as pd
df = pd.read_csv('MSFT.csv')
df.dropna(inplace=True)
# reset the drop missing value
import pandas as pd
df = pd.read_csv('MSFT.csv')
df.reset_index(drop=True, inplace=True)
# Split the data into training and testing sets
train_data = scaled_data[0:training_data_len, :]
x_train = []
y_train = []
for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])
x_train, y_train = np.array(x_train), np.array(y_train)

print(x_train)
print(y_train)
```

Output:

```
[[4.32567884e-05 6.48851826e-05 7.57056091e-05 ... 1.67632514e-04
  1.78446711e-04 1.78446711e-04]
 [6.48851826e-05 7.57056091e-05 5.94780840e-05 ... 1.78446711e-04
  1.78446711e-04 1.45997890e-04]
 [7.57056091e-05 5.94780840e-05 4.86638869e-05 ... 1.78446711e-04
  1.45997890e-04 1.45997890e-04]
 ...
 [1.64827557e-01 1.65824257e-01 1.71493002e-01 ... 1.75417502e-01
  1.74856858e-01 1.74856858e-01]
 [1.65824257e-01 1.71493002e-01 1.69188126e-01 ... 1.74856858e-01
  1.74856858e-01 1.76165034e-01]
 [1.71493002e-01 1.69188126e-01 1.66011144e-01 ... 1.74856858e-01
  1.76165034e-01 1.77660084e-01]]
[1.45997890e-04 1.45997890e-04 1.45997890e-04 ... 1.76165034e-01
 1.77660084e-01 1.77660084e-01]
```

#spliting the data in training

```
x_train, y_train = np.array(x_train), np.array(y_train)

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

print(x_train.shape)
```

Output:

```
(6760, 60, 1)
```

2. Volume:

It help to identify liquidity patterns, that help the trading volumes over time.

3. Moving Averages:

Simple moving averages(SMA) and Exponential Moving Averages(EMA) can help identify trends.

4.Relative Strength Index(RSI):

A momentum oscillator that measures the speed and change of price movements.

5. Moving Average Convergence Divergence(MACD):

Another momentum indicator that can signal changes in trend direction.

MODEL TRAINING:

Once you have prepared a features, we should proceed to model training. Commonly used models for stock price prediction include:

1.Linear Regression:

A simple model that assumes a linear relationship between features and stock prices.

2. Time Series Models:

ARIMA (Auto Regressive Integrated Moving Average), GARCH (Generalized Autoregressive Conditional Heteroskedasticity), or LSTM (Long Short-Term Memory) networks designed to handle sequential data.

3.Machine Learning Models:

Decision Trees, Random Forests, Support Vector Machines, or Gradient Boosting techniques like XGBoost.

4. Deep Learning Models:

Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) for more complex modeling.

5. Ensemble Methods:

Combining multiple models for better predictive performance.

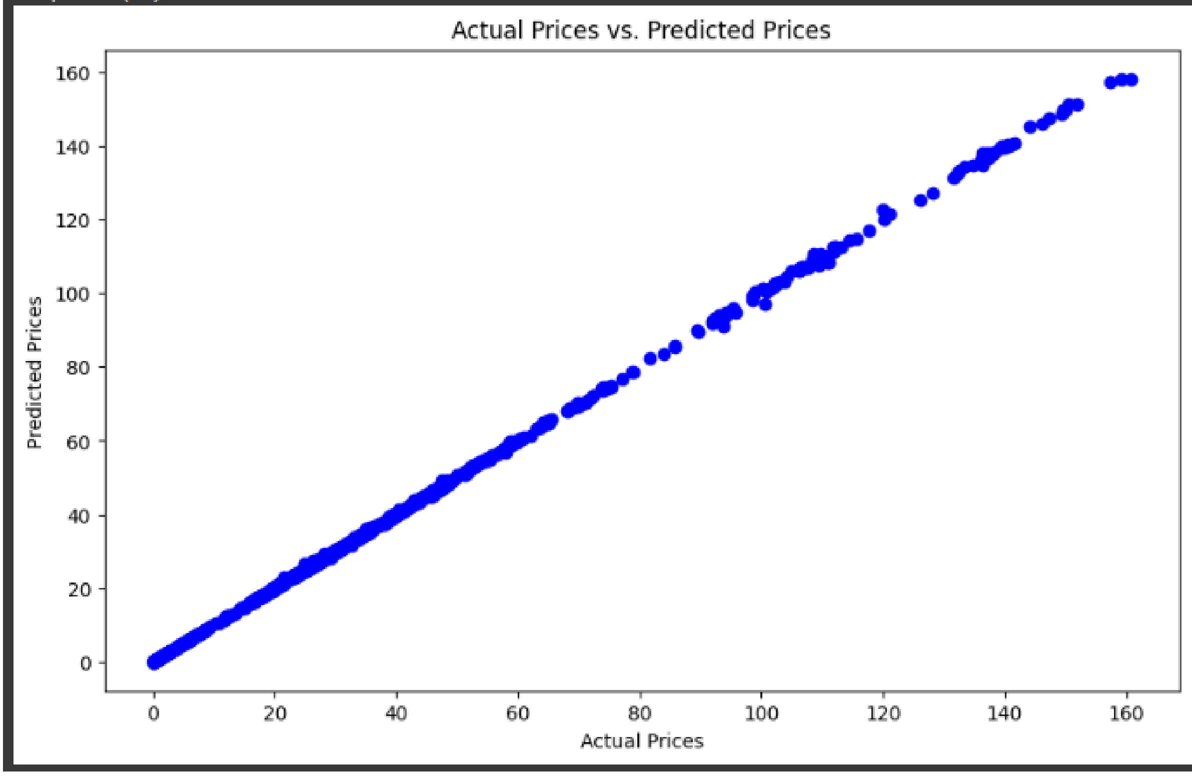
Code:

```
import pandas as pd

import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean_squared_error, r2_score

import matplotlib.pyplot as plt

# Load the dataset

df = pd.read_csv('MSFT.csv')
```

```
#Extract the 'Close' prices as the target variable
target = df['Close']
# Extract relevant features from the dataset (you may need to customize this)
features = df[['Open', 'High', 'Low', 'Volume']]
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=42)
# Initialize and train the Random Forest Regressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)
# Make predictions on the test set
y_pred = rf_regressor.predict(X_test)
# Calculate the model's performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared (R2) Score: {r2}")
# Visualize the actual vs. predicted prices
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue')
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs. Predicted Prices")
plt.show()
```

Output:

```
Mean Squared Error: 0.10332106228204825
R-squared (R2) Score: 0.999879862334485
```



Evaluation:

Above the coding is done for the evaluation part. After training your model, it's essential to evaluate its performance. Common evaluation metrics for regression tasks, such as stock price prediction, include:

1. Mean Absolute Error (MAE): The average absolute difference between predicted and actual values.

2. Mean Squared Error (MSE): The average of the squared differences between predicted and actual values.

3. Root Mean Squared Error (RMSE): The square root of MSE, providing a more interpretable metric.

4. R-squared (R^2): A measure of how well your model explains the variance in the data.

5. Mean Absolute Percentage Error (MAPE): A percentage-based metric, useful for understanding prediction accuracy relative to the stock price.

CONCLUSION:

There are three essential components of stock price prediction: Feature Engineering and Model Training and Evaluation. The code are explained with the three essential components. In this phase of the project, you will further enhance your stock price prediction model through feature engineering, followed by model training and evaluation. These crucial steps will refine the accuracy and effectiveness of your model, bringing you closer to achieving your predictive goals.