IBM NAAN MUDHALVAN

DOMAIN:	Applied Data Science
PROJECT TITLE:	Stock Price Prediction
TEAM MEMBERS:	Ganesh N (420421104019) Sanjay p (420421104067) Tamilselvan A (420421104083) Guganesh R (420421104025) Serinraj M (420421104069)

Dataset link:

https://www.kaggle.com/datasets/prasoonkottarathil/microsoft-lifetime-stocks-dataset

1.Problem Definition:

The problem is to build a predictive model that forecasts stock prices based on historical market data. The goal is to create a tool that assists investors in making well-informed decisions and optimizing their investment strategies. This project involves data collection, data pre-processing, feature engineering, model selection, training, and evaluation.

2.Design Thinking:

1. Data Collection: Collect historical stock market data, including features like date, open price, close price, volume, and other relevant indicators.

2. Data Pre-processing: Clean and pre-process the data, handle missing values, and convert categorical features into numerical representations.

3. Feature Engineering: Create additional features that could enhance the predictive power of the model, such as moving averages, technical indicators, and lagged variables.

4. Model Selection: Choose suitable algorithms for time series forecasting (e.g., ARIMA, LSTM) to predict stock prices.

5. Model Training: Train the selected model using the pre-processed data.

6. Evaluation: Evaluate the model's performance using appropriate time series forecasting metrics (e.g., Mean Absolute Error, Root Mean Squared Error).

3.Algorithm:

1. Start
2. Data collection
3. Data pre-processing
4. Feature Selection
5. Model selection
6. Model training
7. Model evaluation
8. Hyperparameter tuning
9. Deployment and monitoring
10.stop

4.Innovation of the stock price prediction:

In recent years, stock price prediction has seen several innovations driven by advancements in technology and data analysis techniques. These innovations have improved the accuracy and sophistication of stock price forecasting. Here are some notable innovations in stock price prediction:

- **Big Data and Alternative Data Sources:**

The proliferation of big data has allowed analysts to access and analyze vast amounts of information. Alternative data sources, such as social mediasentiment, satellite imagery, and web scraping, have become valuable for predicting stock prices. These non-traditional data streams provide additional insights into market sentiment and company performance.

- **Machine Learning and Deep Learning:**

Machine learning algorithms, including neural networks, have gained popularity in stock price prediction. Deep learning models like recurrent neural networks (RNNs) and long short-term memory (LSTM) networks are effective at handling time series data, making them well-suited for financial forecasting tasks. These models can capture complex patterns and relationships in historical stock price data.

- **Natural Language Processing (NLP):**

NLP techniques are used to analyze news articles, earnings reports, and financial statements. Sentiment analysis of news and social media content helps gauge market sentiment and investor sentiment, which can impact stock prices. NLP can also be used to extract valuable information from unstructured text data.

- **Reinforcement Learning:**

Reinforcement learning algorithms, such as Q-learning and deep reinforcement learning, have been applied to stock trading strategies. These algorithms learn optimal trading decisions through trial and error, adapting to changing market conditions over time.

- **Quantitative Models:**

Quantitative analysts continue to develop and refine mathematical models for stock price prediction. These models may incorporate factors such as volatility, trading volume, and historical price patterns to make predictions.

- **High-Frequency Trading (HFT):**

High-frequency trading firms use advanced algorithms and ultra-fast data access to execute trades within milliseconds. These firms rely on predictive models and real-time data to identify arbitrage opportunities and profit from short-term price movements.

- **Robo-Advisors:**

Robo-advisors leverage automated algorithms to provide investment advice and portfolio management to individual investors. These platforms use predictive models to make asset allocation and trading decisions based on investors' risk profiles and financial goals.

- **Blockchain and Cryptocurrencies:**

In the realm of cryptocurrency markets, blockchain technology has introduced new ways of predicting asset prices. Some predictive models incorporate on-chain data, transaction volume, and network activity to forecast cryptocurrency prices.

- **Explainable AI:**

As AI and machine learning models become more complex, there is a growing emphasis on developing explainable AI solutions. These models aim to provide clear explanations of their predictions, which can be important for gaining trust in financial applications.

- **Quantum Computing (Emerging):**

Although still in the early stages of development, quantum computing holds the potential to revolutionize stock price prediction by significantly increasing computational power. Quantum algorithms may enable the analysis of complex financial models and large datasets at unprecedented speeds.

5.Steps for stock price prediction:

Stock price prediction involves a series of steps that typically follow a systematic process. While there are various methods and techniques used, the following steps provide a general framework for conducting stock price prediction:

1. Data Collection:

Gather historical stock price data for the target stock or asset. This data should include daily, weekly, or intraday prices, trading volumes, and other relevant financial indicators.

2. Feature Selection and Engineering:

Identify and select relevant features (variables) that can impact stock prices. Common features include price indicators (e.g., moving averages, relative strength index), trading
volumes, economic indicators (e.g., GDP growth, inflation rates), news sentiment, and more.

Engineer new features or transform existing ones to capture meaningful patterns or
relationships in the data.

3. Data Pre-processing:

Handle missing data, outliers, and anomalies through imputation or removal.

Normalize or scale features to ensure that they have the same magnitude.

Split the dataset into training, validation, and test sets to evaluate the model's performance.

4. Model Selection:

Choose an appropriate predictive model or algorithm. Common choices include:

Time Series Models (e.g., ARIMA, GARCH): Suitable for modeling stock price movements
over time.

Machine Learning Models (e.g., regression, decision trees, random forests): Can capture
complex relationships between features and stock prices.

Deep Learning Models (e.g., LSTM, CNN): Effective for handling sequential and
time-dependent data.

Ensemble Methods (e.g., Gradient Boosting): Combine multiple models to improve
predictive accuracy.

Consider the trade-offs between model complexity and interpretability.

5. Training the Model:

Use the training dataset to train the chosen model. The model learns the underlying patterns and relationships between features and stock prices during this phase

6. Hyperparameter Tuning:

Optimize the model's hyperparameters through techniques like grid search or random search.
This step helps fine-tune the model's performance.

7. Model Evaluation:

Assess the model's performance using the validation dataset. Common evaluation metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and others.

Consider other metrics such as R-squared (R^2) to measure the model's explanatory power.

Visualize the model's predictions against actual stock prices to gain insights into its
performance.

8. Model Testing:

Evaluate the model's performance on the test dataset, which represents unseen data. This step helps assess the model's ability to generalize to new data.

9. Model Interpretation (Optional):

If using interpretable models, analyse the model's coefficients or feature importance scores to understand which factors have the most significant impact on stock price predictions.

10. Deployment (Optional):

In some cases, particularly for trading algorithms or investment strategies, the predictive model may be deployed in a live trading environment.

11. Monitoring and Updating:

Continuously monitor the model's performance in real-world conditions and update it as needed to adapt to changing market dynamics.

12. Risk Management:

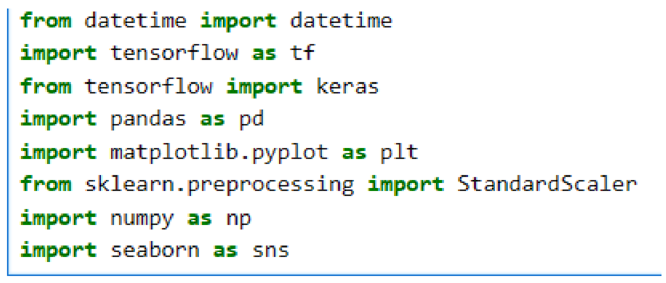
Implement risk management strategies to mitigate potential losses. This may include setting stop-loss orders, diversifying the portfolio, or using position sizing techniques.

13. Regulatory Compliance (if applicable):

Ensure compliance with financial regulations and trading rules, especially if the prediction model is used for trading purposes.

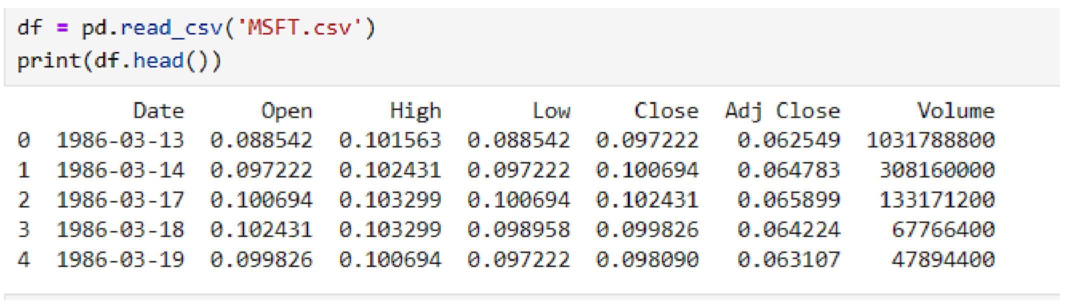
6.Dataset Loading and Describe:

- **Import the Libraries:**



```
from datetime import datetime
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
import numpy as np
import seaborn as sns
```

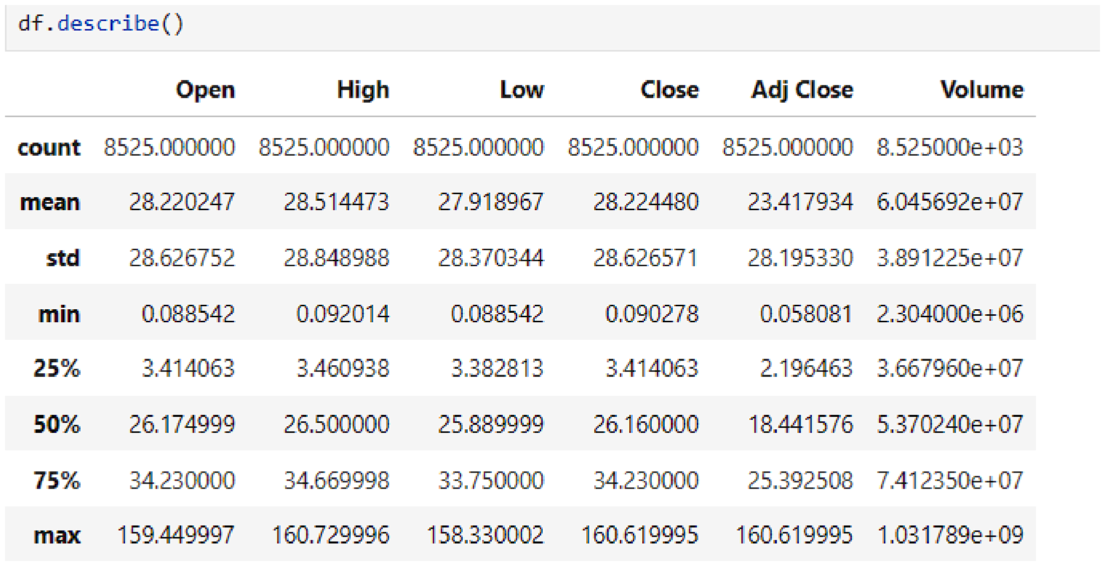
- **Import the dataset:**



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

```
         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
```

- **Describe the dataset:**



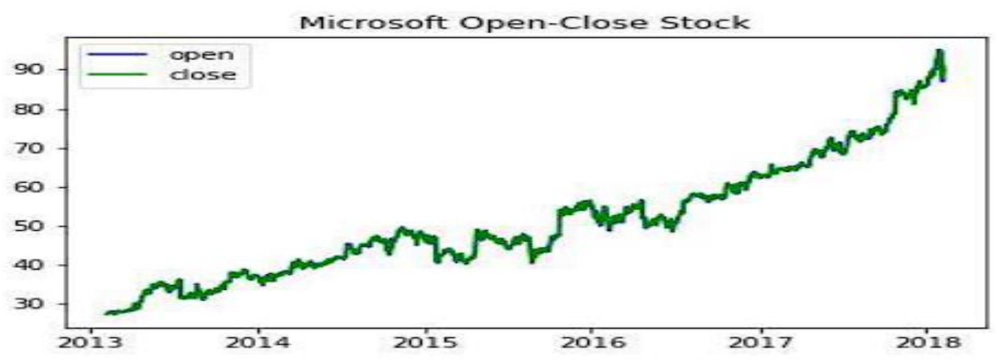
```
df.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	8525.000000	8525.000000	8525.000000	8525.000000	8525.000000	8.525000e+03
mean	28.220247	28.514473	27.918967	28.224480	23.417934	6.045692e+07
std	28.626752	28.848988	28.370344	28.626571	28.195330	3.891225e+07
min	0.088542	0.092014	0.088542	0.090278	0.058081	2.304000e+06
25%	3.414063	3.460938	3.382813	3.414063	2.196463	3.667960e+07
50%	26.174999	26.500000	25.889999	26.160000	18.441576	5.370240e+07
75%	34.230000	34.669998	33.750000	34.230000	25.392508	7.412350e+07
max	159.449997	160.729996	158.330002	160.619995	160.619995	1.031789e+09

7.Data Visualization:

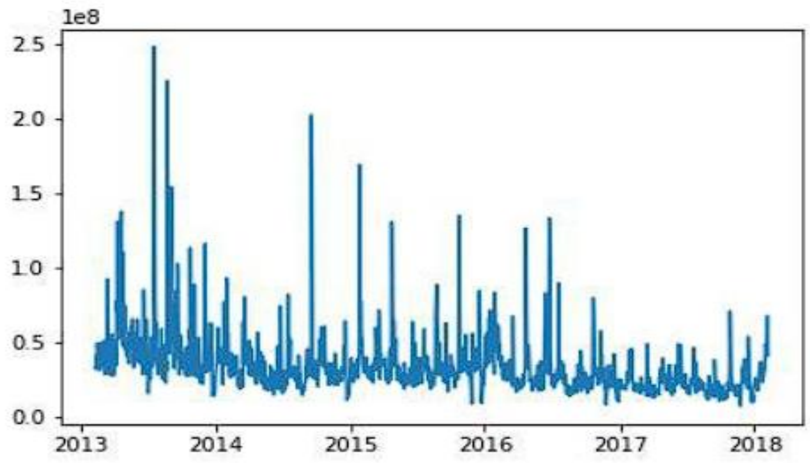
```
plt.plot(df['date'],
         df['open'],
         color="blue",
         label="open")
plt.plot(df['date'],
         df['close'],
         color="green",
         label="close")
plt.title("Microsoft Open-Close Stock")
plt.legend()
```

Output



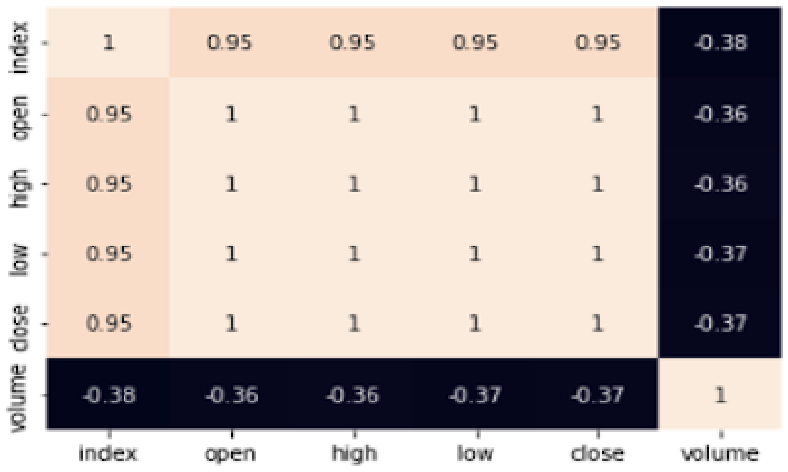
```
plt.plot(df['date'],
         df['volume'])
plt.show()
```

Output:



```
sns.heatmap(df.corr(),
            annot=True,
            cbar=False)
plt.show()
```

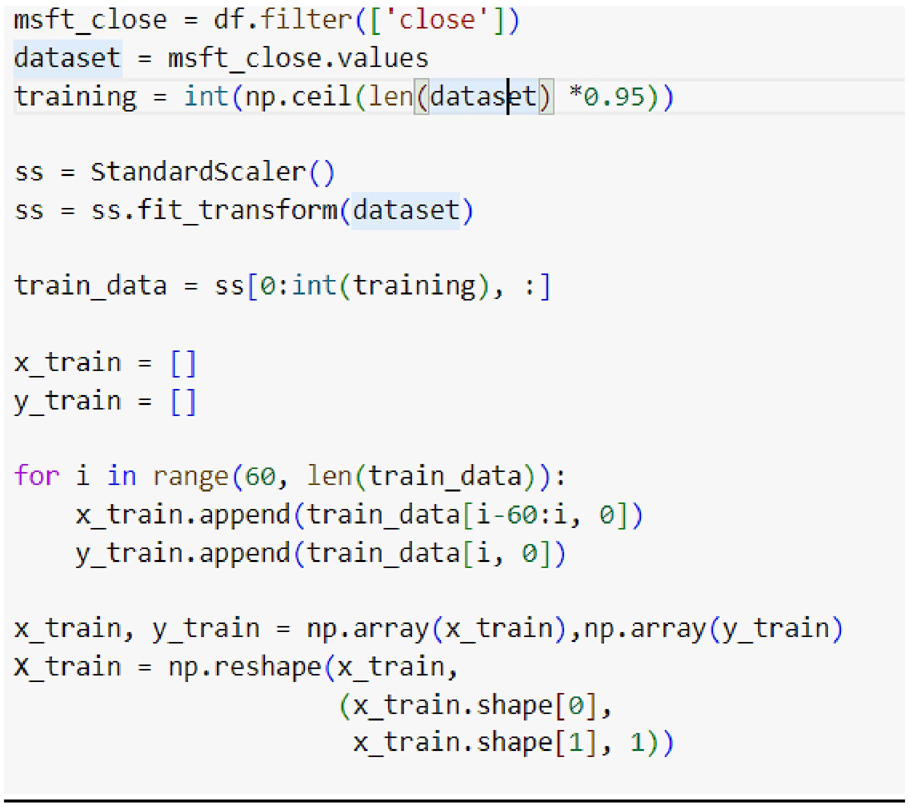
Output:



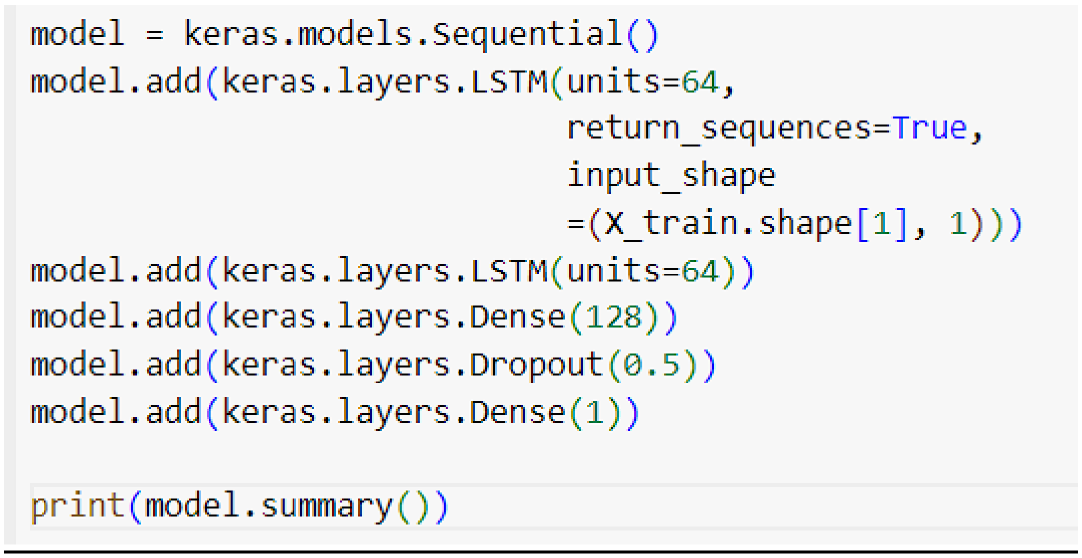
	index	open	high	low	close	volume
index	1	0.95	0.95	0.95	0.95	-0.38
open	0.95	1	1	1	1	-0.36
high	0.95	1	1	1	1	-0.36
low	0.95	1	1	1	1	-0.37
close	0.95	1	1	1	1	-0.37
volume	-0.38	-0.36	-0.36	-0.37	-0.37	1

8.Build the model:

- **Create the X_Train and Y_Train:**

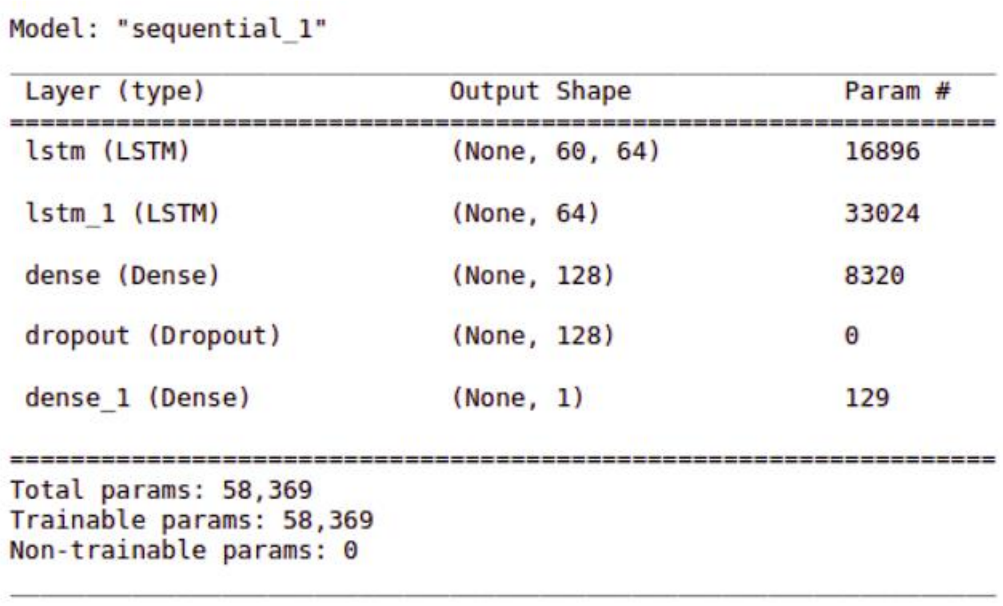


```
msft_close = df.filter(['close'])
dataset = msft_close.values
training = int(np.ceil(len(dataset) *0.95))

ss = StandardScaler()
ss = ss.fit_transform(dataset)

train_data = ss[0:int(training), :]

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

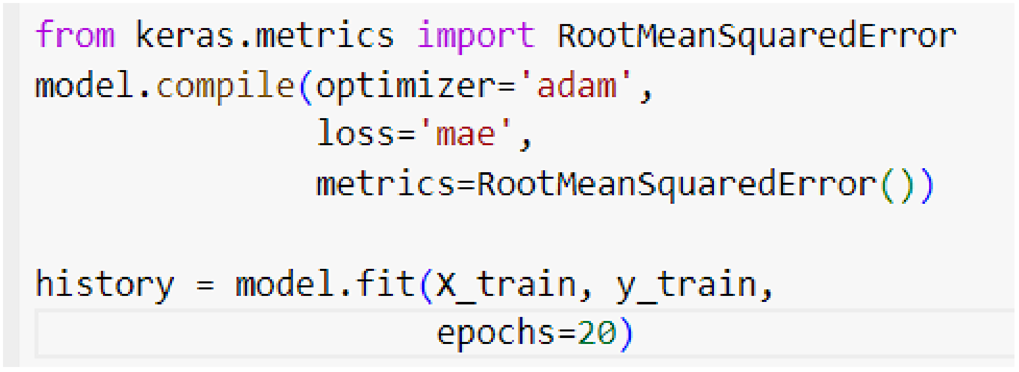


```
model = keras.models.Sequential()
model.add(keras.layers.LSTM(units=64,
                            return_sequences=True,
                            input_shape
                            =(X_train.shape[1], 1)))
model.add(keras.layers.LSTM(units=64))
model.add(keras.layers.Dense(128))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(1))

print(model.summary())
```

Output:

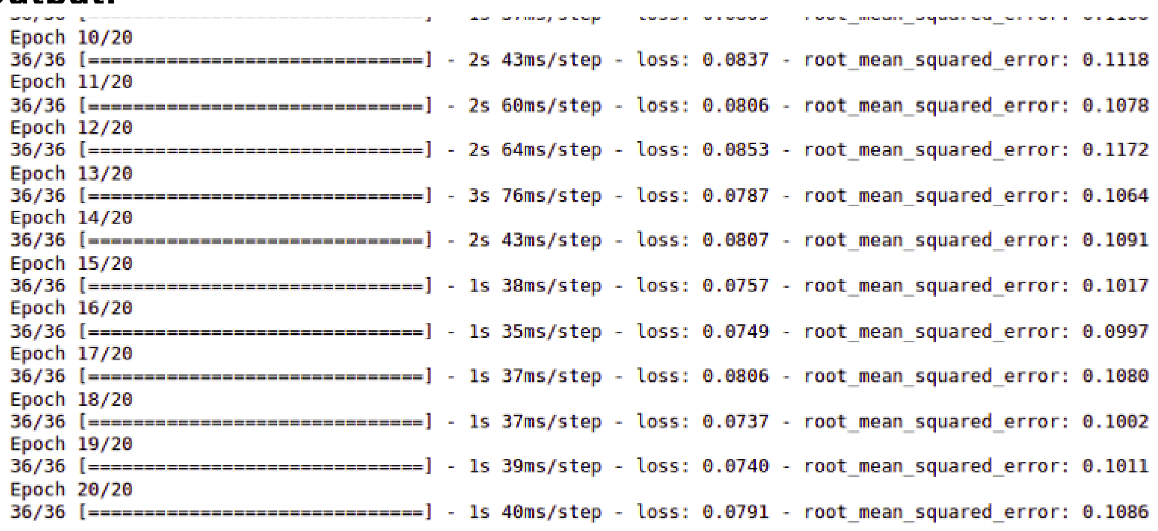


```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 64)            16896

 lstm_1 (LSTM)               (None, 64)                33024

 dense (Dense)               (None, 128)               8320

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 58,369
Trainable params: 58,369
Non-trainable params: 0
_________________________________________________________________
```



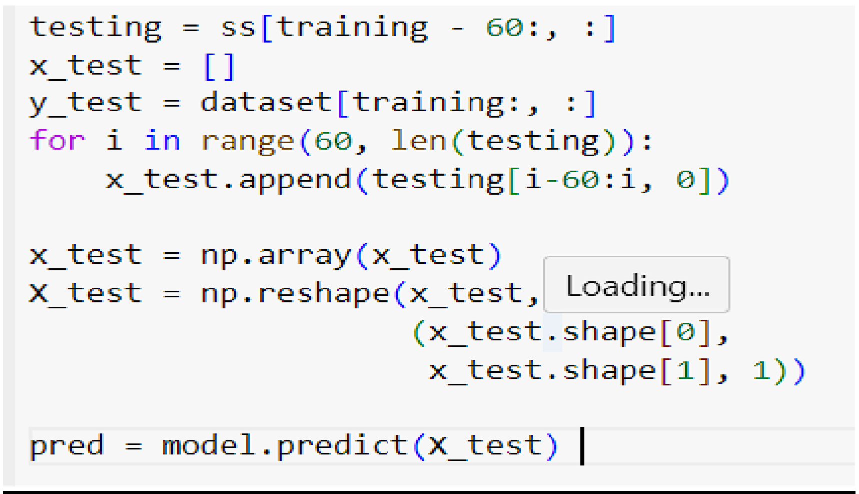
```python
from keras.metrics import RootMeanSquaredError
model.compile(optimizer='adam',
              loss='mae',
              metrics=RootMeanSquaredError())

history = model.fit(X_train, y_train,
                    epochs=20)
```

Output:



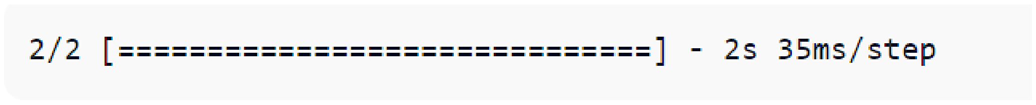
```
Epoch 10/20
36/36 [==============================] - 2s 43ms/step - loss: 0.0837 - root_mean_squared_error: 0.1118
Epoch 11/20
36/36 [==============================] - 2s 60ms/step - loss: 0.0806 - root_mean_squared_error: 0.1078
Epoch 12/20
36/36 [==============================] - 2s 64ms/step - loss: 0.0853 - root_mean_squared_error: 0.1172
Epoch 13/20
36/36 [==============================] - 3s 76ms/step - loss: 0.0787 - root_mean_squared_error: 0.1064
Epoch 14/20
36/36 [==============================] - 2s 43ms/step - loss: 0.0807 - root_mean_squared_error: 0.1091
Epoch 15/20
36/36 [==============================] - 1s 38ms/step - loss: 0.0757 - root_mean_squared_error: 0.1017
Epoch 16/20
36/36 [==============================] - 1s 35ms/step - loss: 0.0749 - root_mean_squared_error: 0.0997
Epoch 17/20
36/36 [==============================] - 1s 37ms/step - loss: 0.0806 - root_mean_squared_error: 0.1080
Epoch 18/20
36/36 [==============================] - 1s 37ms/step - loss: 0.0737 - root_mean_squared_error: 0.1002
Epoch 19/20
36/36 [==============================] - 1s 39ms/step - loss: 0.0740 - root_mean_squared_error: 0.1011
Epoch 20/20
36/36 [==============================] - 1s 40ms/step - loss: 0.0791 - root_mean_squared_error: 0.1086
```

9.Model Evaluation:



```
testing = ss[training - 60:, :]
x_test = []
y_test = dataset[training:, :]
for i in range(60, len(testing)):
    x_test.append(testing[i-60:i, 0])

x_test = np.array(x_test)
X_test = np.reshape(x_test,
                    (x_test.shape[0],
                     x_test.shape[1], 1))

pred = model.predict(X_test)
```

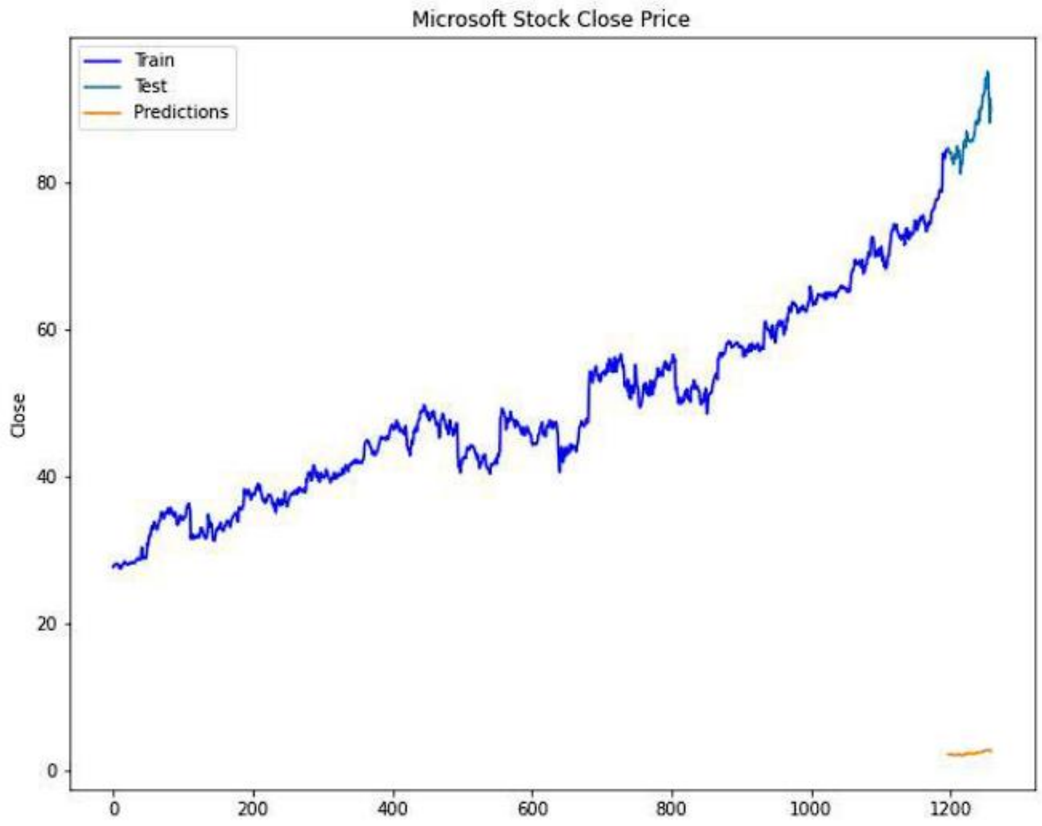
Output:



```
2/2 [==============================] - 2s 35ms/step
```

```
train =df[:training]
test = df[training:]
test['Predictions'] = pred

plt.figure(figsize=(10, 8))
plt.plot(train['close'], c="b")
plt.plot(test[['close', 'Predictions']])
plt.title('Microsoft Stock Close Price')
plt.ylabel("Close")
plt.legend(['Train', 'Test', 'Predictions'])
```

Output:



<u>10.Insights Gained:</u>

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange. The successful prediction of a stock's future price could yield significant profit.

The efficient-market hypothesis suggests that stock prices reflect all currently available information and any price changes that are not based on newly revealed information thus are inherently unpredictable.

Others disagree and those with this viewpoint possess myriad methods and technologies which purportedly allow them to gain future price information.

<u>11.conclusion:</u>

stock price prediction is a challenging endeavor that combines various analytical techniques and approaches. It involves analysing historical data, considering fundamental and technical indicators, monitoring market sentiment, and using machine learning models.