Stock Market Prediction Using Machine Learning

By : Atharv Patwardhan

(Date : 16-05-2022)

1. **Problem Statement :**

   A stock market is a public market where people can buy and sell shares of listed companies. On buying these stocks, a person owns a percentage of that company.

   The Stock market is one of the most widely used platforms for people to invest their money.

   Unfortunately, many people don't have much knowledge about stocks and tend to invest blindly without prior study and analysis. This results in losses.

2. **Customer Need Assesment :**

   Shockingly, it is estimated that almost 90% of people lose money in the stock market. This includes both new and experienced investors. This is largely because they tend to invest based on rumours and speculations. Someone gives them a stock tip and they invest in that stock blindly. There are many stock frauds happening as well. These are usually conducted over SMS where the victims are sent messages such as "Buy 1000 stocks of XYZ company for a discounted price!","XYZ company stock at it's lowest, buy now!" etc.

3. **Target Specification and Characterization :**

Most people who fall for stock fraud are between the ages 60 to 69. People aged 30 to 39 were the second most commonly targeted group. But scams are just a

Part of the problem. Many people from all age groups invest blindly lose their Because of their own mistakes.This product will help people in analysing stocks and making decisions based on facts and not just tips and speculations.

4. External References :

The dataset is taken from yahoo finance :

https://finance.yahoo.com/quote/AAPL/history/

The model is based on the model explained in this video :
https://www.youtube.com/watch?v=QIUxPv5PJOY

5. Benchmarking :

The model is not completely flawless in calculating the stock prices but accuracy improves after training the model. The predicted stock price differs from the actual price only by a couple of cents.

6. Applicable Patents :

No patents are applicable to this as the project is open source.

7. Applicable Regulations :

- SEBI stock rules
- International Stock Trading Rules

8. **Applicable Constraints :**
1. Machine Learning Specialist : To help find and fix flaws (if any) in the algorithm and code
2. Data Scientist : To help make conclusions from the data and check if the model is giving comparable conclusions.
3. Stock Market Specialists : To help make sure the product is abiding all regulations on stock trading.
4. DevOps Engineers : To make sure all the servers and network are running smoothly.

9. **Business Opportunity :**

The stock market is an ever growing market with thousands or people joining it everyday. Hence the product has an ever growing scope as more and more people want to invest their money in stocks. In 2021 itself, there were over 142 Lakh new investors.

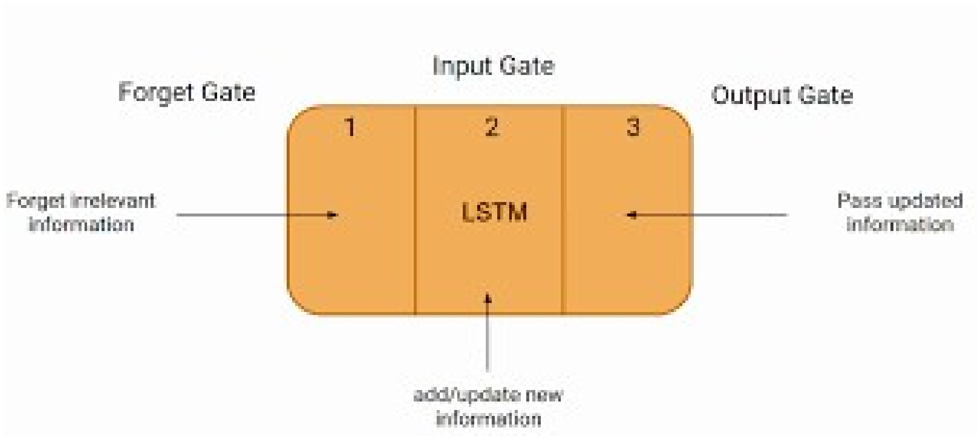
10. **Concept Generation :**

Millions of people lose money in the stock market due to lack of knowledge or analysis. Some people even invest their lifetime savings into the market just because someone told them to or they got scammed.

This product will help people do their own anaylsis and become more aware of stock market trends.

11.Concept Development :

For this project, we have used the LSTM (Long Short Term Memory) algorithm. The LSTM algorithm is an algorithm that is well suited for making predictions based on time and series data. We use it here because we need to make predictions based on the closing price of stocks in the past.



12.Final Product Prototype:

Back End :

- All the Data Engineers, Backend developers and analysts work here.
- It is used to store,process and analyze data.
- The model should constantly be updated with the new datasets.

Front End :

- All the software developers and Web/UI developers work in this area.
- There should be a good user interface present to make customer navigation easier.
- It should have an attractive UI to attract more people.

13.Product Details:

Working :

i. Firstly, the user is asked to select which company/stock he or she wants to analyze.
ii. Then, the stock data is fetched from yahoo finance.
iii. All the fetched data is given to the model to process.
iv. After the data is run through the algorithm, the machine predicts a closing price for the stock.
v. The model uses the LSTM algorithm for processing.

Data Source :

The Data is sourced from yahoo finance as mentioned above.

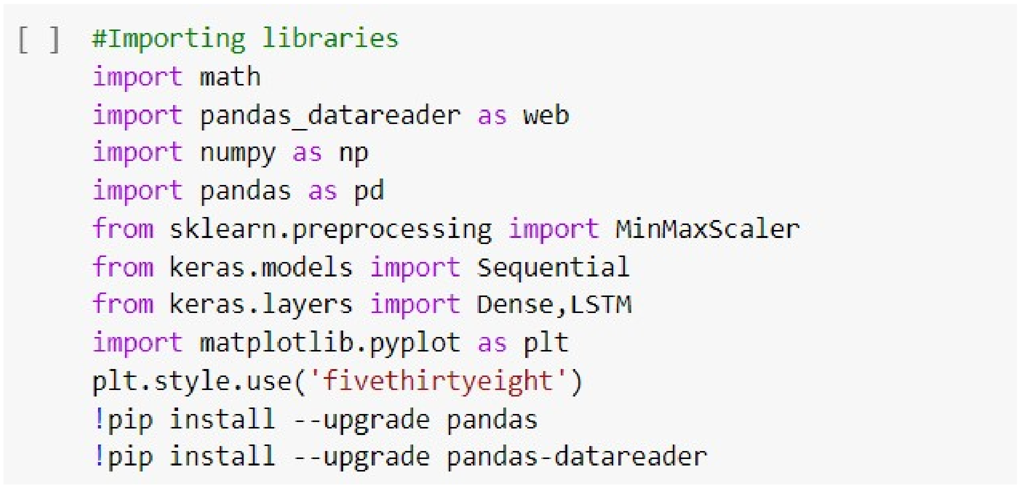
https://finance.yahoo.com/

For the demonstration, we have used data of the stocks from Apple inc.

14.Code Implementation

Here is some of the code implementation :

Importing Required Libraries and Modules



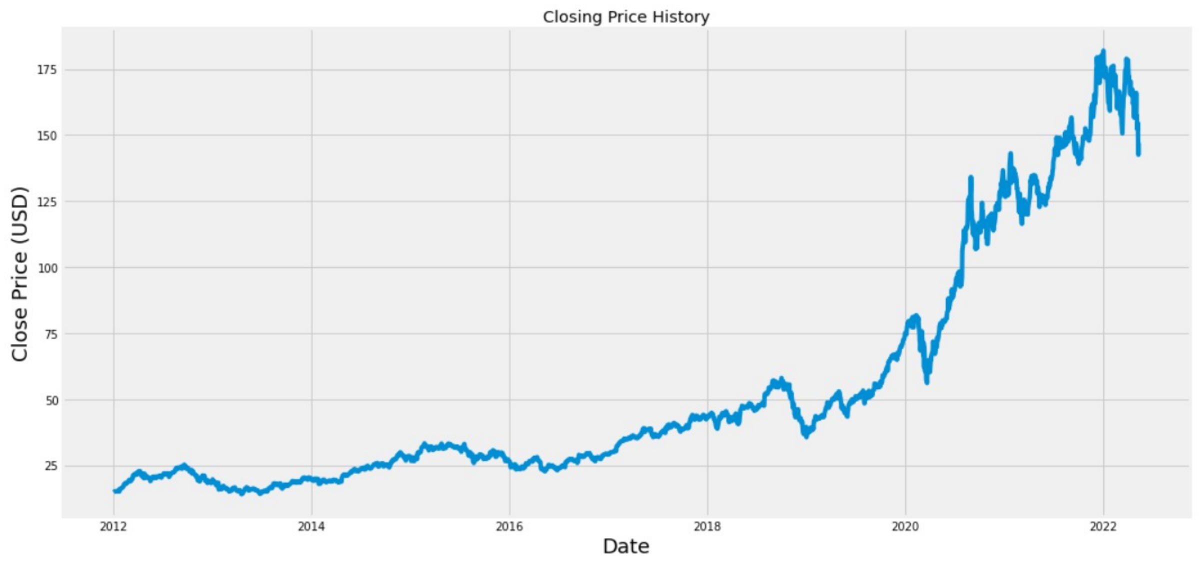
```
[ ]  #Importing libraries
     import math
     import pandas_datareader as web
     import numpy as np
     import pandas as pd
     from sklearn.preprocessing import MinMaxScaler
     from keras.models import Sequential
     from keras.layers import Dense,LSTM
     import matplotlib.pyplot as plt
     plt.style.use('fivethirtyeight')
     !pip install --upgrade pandas
     !pip install --upgrade pandas-datareader
```

Getting the dataset

```
#Get the stock quotes
df = web.DataReader('AAPL',data_source='yahoo',start='2012-01-01',end='2022-05-13')
#Show the data
df
```

Visualizing the closing price history

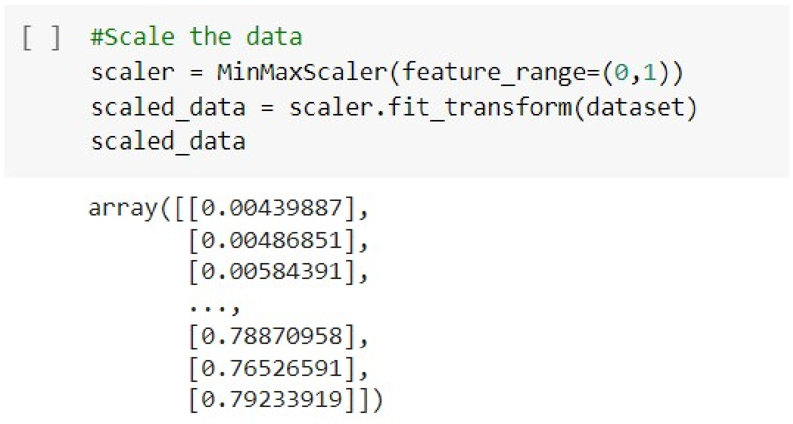
```
[ ] #Visualize closing price history
    plt.figure(figsize=(16,8))
    plt.title('Closing Price History')
    plt.plot(df['Close'])
    plt.xlabel('Date',fontsize=18)
    plt.ylabel('Close Price (USD)',fontsize=18)
    plt.show()
```



Filtering out the data and keeping only the closing price column in the dataset

```
[ ] #Create a new dataframe with only the Close Column
    data = df.filter(['Close'])
    #Convert dataframe to a numpy array
    dataset = data.values
    #Get the number of rows to train the model on
    training_data_len = math.ceil(len(dataset)* .8)
    training_data_len
```

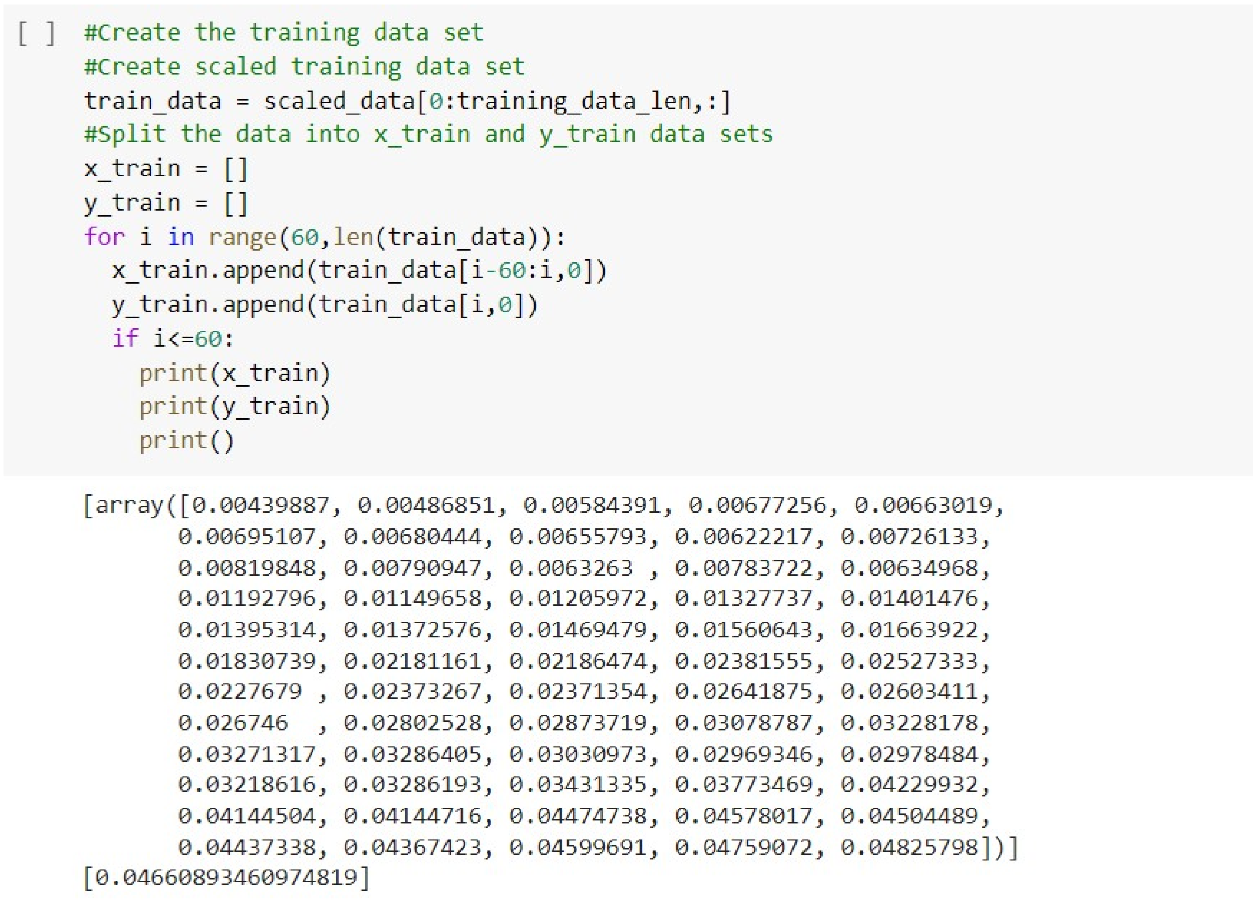
Scaling the data to values between 0 and 1



```
[ ]  #Scale the data
     scaler = MinMaxScaler(feature_range=(0,1))
     scaled_data = scaler.fit_transform(dataset)
     scaled_data
```

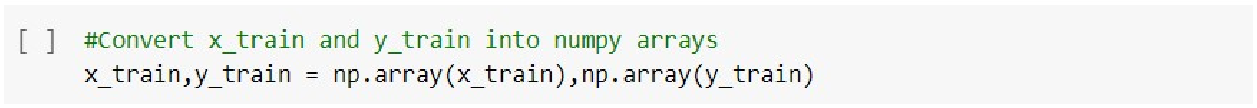
```
array([[0.00439887],
       [0.00486851],
       [0.00584391],
       ...,
       [0.78870958],
       [0.76526591],
       [0.79233919]])
```

Creating the training set and converting it into a numpy array



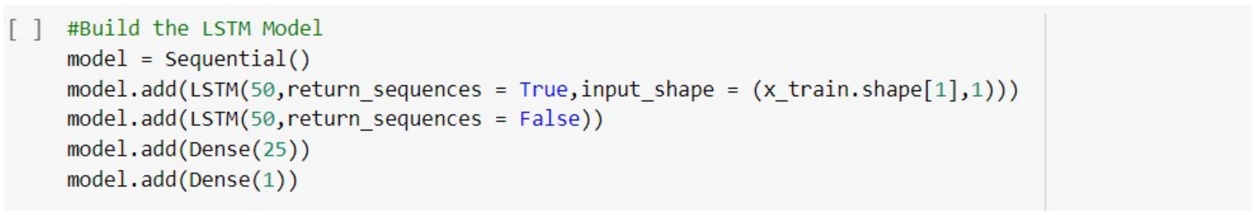
```
[ ]  #Create the training data set
     #Create scaled training data set
     train_data = scaled_data[0:training_data_len,:]
     #Split the data into x_train and y_train data sets
     x_train = []
     y_train = []
     for i in range(60,len(train_data)):
       x_train.append(train_data[i-60:i,0])
       y_train.append(train_data[i,0])
       if i<=60:
         print(x_train)
         print(y_train)
         print()
```

```
[array([0.00439887, 0.00486851, 0.00584391, 0.00677256, 0.00663019,
       0.00695107, 0.00680444, 0.00655793, 0.00622217, 0.00726133,
       0.00819848, 0.00790947, 0.0063263 , 0.00783722, 0.00634968,
       0.01192796, 0.01149658, 0.01205972, 0.01327737, 0.01401476,
       0.01395314, 0.01372576, 0.01469479, 0.01560643, 0.01663922,
       0.01830739, 0.02181161, 0.02186474, 0.02381555, 0.02527333,
       0.0227679 , 0.02373267, 0.02371354, 0.02641875, 0.02603411,
       0.026746  , 0.02802528, 0.02873719, 0.03078787, 0.03228178,
       0.03271317, 0.03286405, 0.03030973, 0.02969346, 0.02978484,
       0.03218616, 0.03286193, 0.03431335, 0.03773469, 0.04229932,
       0.04144504, 0.04144716, 0.04474738, 0.04578017, 0.04504489,
       0.04437338, 0.04367423, 0.04599691, 0.04759072, 0.04825798])]
[0.04660893460974819]
```

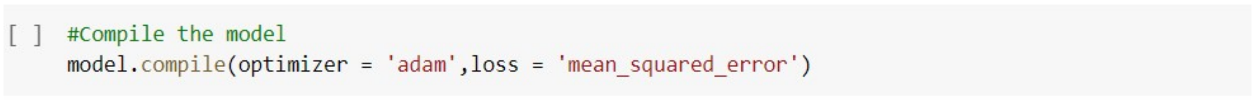


```
[ ]  #Convert x_train and y_train into numpy arrays
     x_train,y_train = np.array(x_train),np.array(y_train)
```

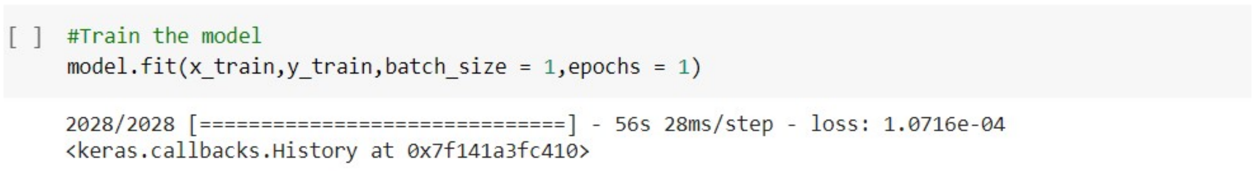
Building and training the model



```
[ ]  #Build the LSTM Model
     model = Sequential()
     model.add(LSTM(50,return_sequences = True,input_shape = (x_train.shape[1],1)))
     model.add(LSTM(50,return_sequences = False))
     model.add(Dense(25))
     model.add(Dense(1))
```



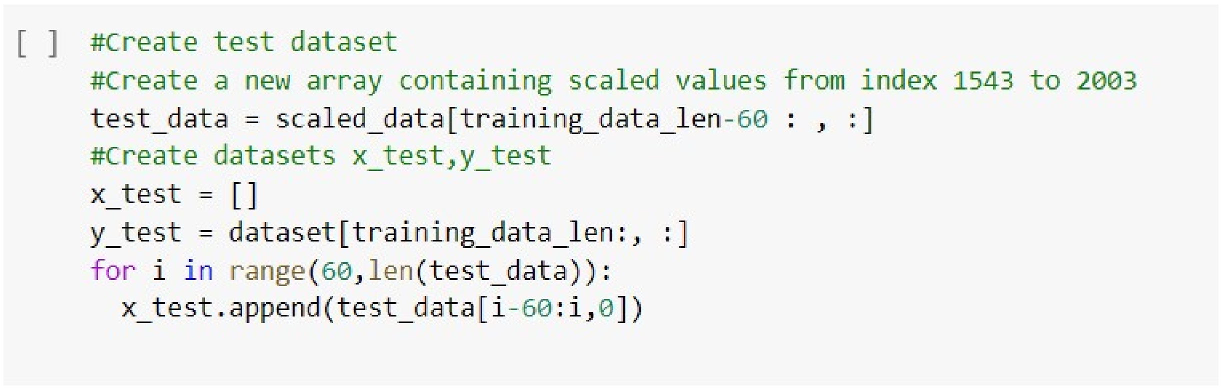
```
[ ]  #Compile the model
     model.compile(optimizer = 'adam',loss = 'mean_squared_error')
```



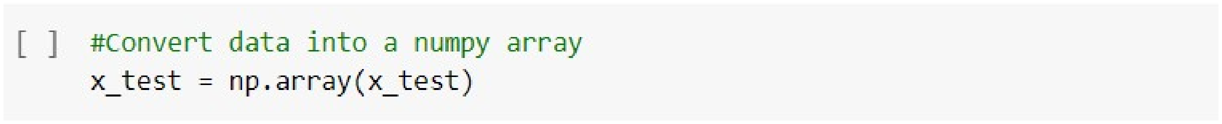
```
[ ]  #Train the model
     model.fit(x_train,y_train,batch_size = 1,epochs = 1)

     2028/2028 [==============================] - 56s 28ms/step - loss: 1.0716e-04
     <keras.callbacks.History at 0x7f141a3fc410>
```

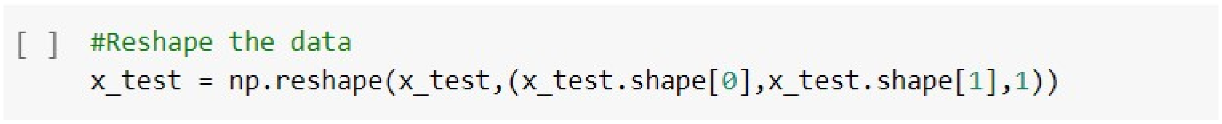
Creating the test dataset



```
[ ]  #Create test dataset
     #Create a new array containing scaled values from index 1543 to 2003
     test_data = scaled_data[training_data_len-60 : , :]
     #Create datasets x_test,y_test
     x_test = []
     y_test = dataset[training_data_len:, :]
     for i in range(60,len(test_data)):
       x_test.append(test_data[i-60:i,0])
```

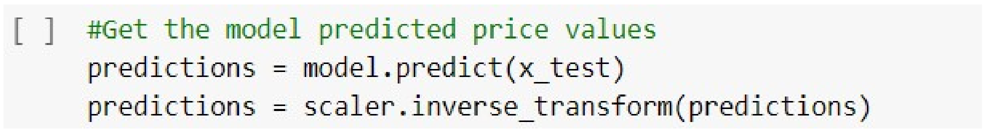


```
[ ]  #Convert data into a numpy array
     x_test = np.array(x_test)
```



```
[ ]  #Reshape the data
     x_test = np.reshape(x_test,(x_test.shape[0],x_test.shape[1],1))
```

Get the predicted values



```
[ ]  #Get the model predicted price values
     predictions = model.predict(x_test)
     predictions = scaler.inverse_transform(predictions)
```

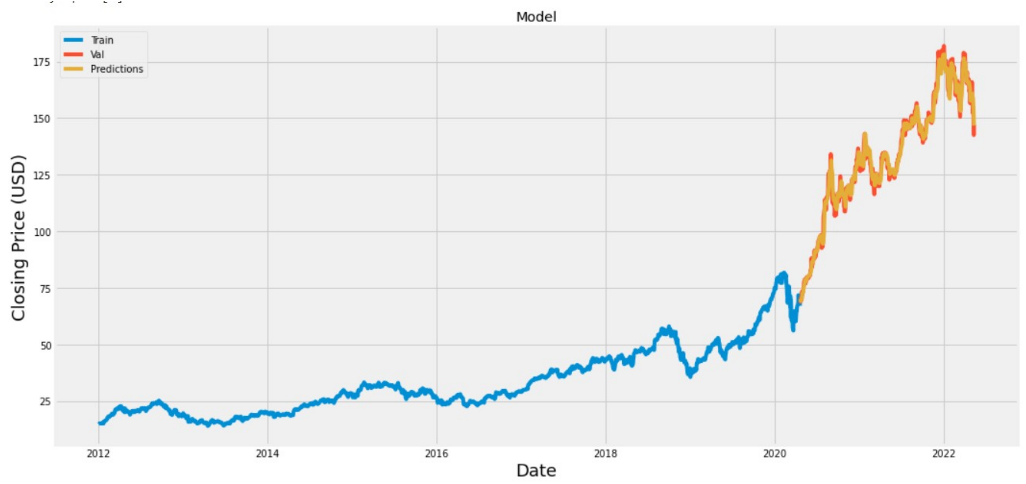
Find the Root Mean Square Error

```
[ ]  #Get the Root Mean Squared Error (RMSE)
     rmse = np.sqrt(np.mean(predictions - y_test)**2)
     rmse
```

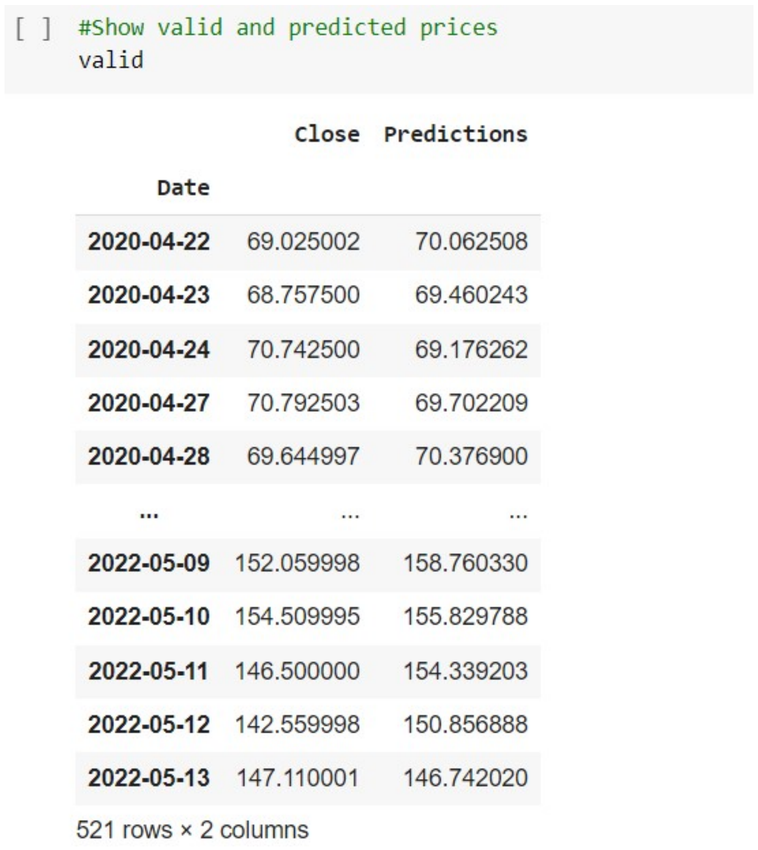
```
0.18060856268181683
```

Plot the data

```
[ ]  #Plot the data
     train = data[: training_data_len]
     valid = data[training_data_len :]
     valid['Predictions'] = predictions
     #Visualize the data
     plt.figure(figsize = (16,8))
     plt.title('Model')
     plt.xlabel('Date',fontsize=18)
     plt.ylabel('Closing Price (USD)',fontsize = 18)
     plt.plot(train['Close'])
     plt.plot(valid[['Close','Predictions']])
     plt.legend(['Train','Val','Predictions'],loc = 'lower:right')
     plt.show()
```



Get the predicted and actual prices for comparison



```
[ ]  #Show valid and predicted prices
     valid
```

	Close	Predictions
Date		
2020-04-22	69.025002	70.062508
2020-04-23	68.757500	69.460243
2020-04-24	70.742500	69.176262
2020-04-27	70.792503	69.702209
2020-04-28	69.644997	70.376900
...	...	...
2022-05-09	152.059998	158.760330
2022-05-10	154.509995	155.829788
2022-05-11	146.500000	154.339203
2022-05-12	142.559998	150.856888
2022-05-13	147.110001	146.742020

521 rows × 2 columns

15.Conclusion

This model predicts closing prices of stocks quite accurately and can be optimised even more by professionals with more technical knowledge. This model was built with the aim of helping people invest their money more wisely and it is capable of serving it's purpose.

References :

https://en.wikipedia.org/wiki/Long_short-term_memory#:~:text=LSTM%20networks%20are%20well%2Dsuited,encountered%20when%20training%20traditional%20RNNs.

https://www.youtube.com/watch?v=QIUxPv5PJOY

https://finance.yahoo.com/

Github Link : https://github.com/atharvpatwardhan/StockMarketPrediction