# Open Ended Problem

## AIM: Implement Stock Exchange Prediction Small Project with Charts in Python.

```
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.python.keras.layers import Dense,LSTM
from tensorflow.python.keras import Sequential
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
#get the stock quote

df=web.DataReader('TSLA',data_source='yahoo',start='2010-01-01',end='2019-12-17')
df

#visualize the closing price hystpl
plt.figure(figsize=(16,8))
plt.title('Close price History')
plt.plot(df['Close'])
plt.xlabel('Date',fontsize=18)
plt.ylabel('Close Price USD ($) ',fontsize=18)
plt.show()

#get the numbers of rows and columns
df.shape

#create a new dataframe with only the 'Close column
```

```
data=df.filter(['Close'])
#convert the dataframe to a numpy array
dataset=data.values
print(len(dataset))

#get the number of rowa to train the model on [HAS ISSUE WITH LENGTH]
training_data_len=math.ceil(len(dataset) * .8)
training_data_len

#Scale the data
scaler= MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(dataset)
print(len(scaled_data))
scaled_data

#create the training dataset
#create the scaed training data set

train_data=scaled_data[0:training_data_len,:] #0-1908 [issue here]

#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])
print(len(x_train))
print()
print(len(y_train))
```

```
#Convert the x_train and y_train to numpy arrays
x_train,y_train = np.array(x_train),np.array(y_train)

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

#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 model
model.compile(optimizer='adam',loss='mean_squared_error')

#train the model
model.fit(x_train,y_train,batch_size=1,epochs=1)

#create the testing dataset
#create a new array containing scaled value
test_data=scaled_data[training_data_len-60:,:]

#Create the data set x_test and 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 the data into numpy dataset
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 model predicted price values
predictions=model.predict(x_test)
predictions=scaler.inverse_transform(predictions)

#Get the root mean squared error (RMSE)
rmse=np.sqrt(np.mean(predictions-y_test)**2)
rmse

#Plot the data
train=data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions']=predictions

#visual
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date',fontsize=18)
plt.ylabel('Close Price USD($)',fontsize=18)
plt.plot(train['Close'])
plt.plot(valid[['Close','Predictions']])
plt.legend(['Train','Val','Predictions'],loc='lower right')
plt.show

#Show the valid and predicted prices valid
```

```
#Get the quote
apple_quote = web.DataReader('TSLA',data_source='yahoo', start='2010-01-01' ,end='2020-2-5')

#create a new datafreame new
df =apple_quote.filter(['Close'])

#get the last 60 days values
last_60_days=new_df[-60:].values

#scale the data to be values btn 0-1
last_60_days_scaled = scaler.transform(last_60_days)
X_test= []
X_test.append(last_60_days_scaled)
X_test = np.array(X_test)

#Resahpe
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
pred_price=model.predict(X_test)
pred_price=scaler.inverse_transform(pred_price)
print("Predicted price of given date is : ")
print(pred_price)
```

## OUTPUT:

### Data-of-tesla company:

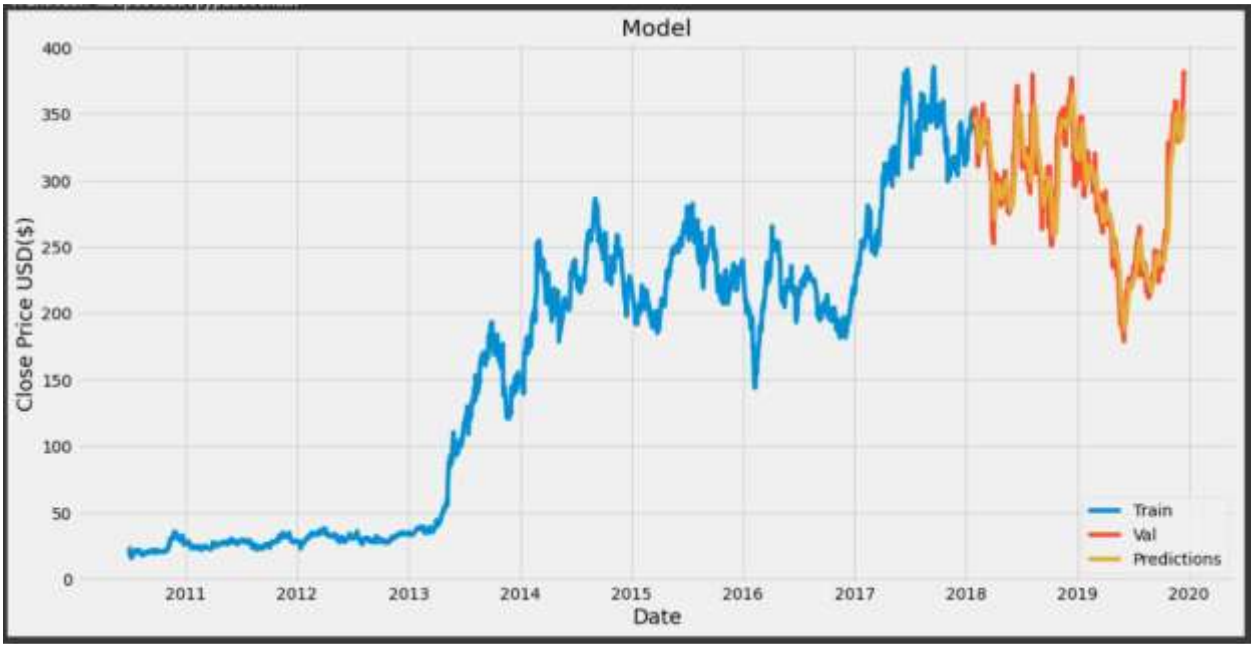
| Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| 2010-06-29 | 25.000000 | 17.540001 | 19.000000 | 23.889999 | 18766300 | 23.889999 |
| 2010-06-30 | 30.420000 | 23.299999 | 25.790001 | 23.830000 | 17187100 | 23.830000 |
| 2010-07-01 | 25.920000 | 20.270000 | 25.000000 | 21.959999 | 8218800 | 21.959999 |
| 2010-07-02 | 23.100000 | 18.709999 | 23.000000 | 19.200001 | 5139800 | 19.200001 |
| 2010-07-06 | 20.000000 | 15.830000 | 20.000000 | 16.110001 | 6866900 | 16.110001 |
| ... | ... | ... | ... | ... | ... | ... |
| 2019-12-11 | 357.190002 | 351.089996 | 351.880005 | 352.700012 | 6897800 | 352.700012 |
| 2019-12-12 | 362.739990 | 353.230011 | 354.920013 | 359.679993 | 7763900 | 359.679993 |
| 2019-12-13 | 365.209991 | 354.640015 | 361.049988 | 358.390015 | 6570900 | 358.390015 |
| 2019-12-16 | 383.609985 | 362.500000 | 362.549988 | 381.500000 | 18174200 | 381.500000 |
| 2019-12-17 | 385.500000 | 375.899994 | 378.989990 | 378.989990 | 8496800 | 378.989990 |

2385 rows × 6 columns

### Training –of –model-LSTM:

```
1848/1848 [==============================] - 47s 26ms/step - loss: 0.0027
<tensorflow.python.keras.callbacks.History at 0x7f51aa4a59e8>
```

Graph-of-prediction-and-training:




Predicted-price for date 2020-04-04:

```
[[492.92722]]
```