Sardar Patel Institute of Technology,Mumbai
Department of Electronics and Telecommunication Engineering
B.E. Sem-VII (2021-2022)
Data Analytics

Experiment: Exploratory Data Analysis (EDA)

Name: Sneha Ghuge **UID:** 2019110015 **BE ETRX** **DA LAB 7**

Aim: Time Series Forecasting: Data, Analysis, and Practice

Problem Statement : To build a regression model and measure the performance of the model in terms of accuracy, recall, sensitivity, specificity, ROC curves, precision recall curves and loss function for regression.

CODE & OUTPUT:

Importing Libraries

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

from tensorflow import keras
from tensorflow.keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, BatchNormalization, Dropout, LSTM, Activation
from tensorflow.keras.models import Sequential
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from tensorflow.keras.utils import to_categorical

from cv2 import imread, resize
```

Importing the data for Microsoft Stock

```
df = pd.read_csv("C:\Users\91744\Desktop\Sem VII\Sem VII Lab\DA Lab\Lab 7 - Time Series\Microsoft_Stock.csv")
df.head()
```

	Date	Open	High	Low	Close	Volume
0	4/1/2015 16:00:00	40.60	40.76	40.31	40.72	36865322
1	4/2/2015 16:00:00	40.66	40.74	40.12	40.29	37487476
2	4/6/2015 16:00:00	40.34	41.78	40.18	41.55	39223692
3	4/7/2015 16:00:00	41.61	41.91	41.31	41.53	28809375
4	4/8/2015 16:00:00	41.48	41.69	41.04	41.42	24753438

Dropping the “Volume” Column bcoz its not needed for the analysis

```
df = df.drop(['Volume'], axis = 1)
df.Date = pd.to_datetime(df['Date'])
df.set_index('Date', inplace = True)

df
```

Date	Open	High	Low	Close
2015-04-01 16:00:00	40.60	40.76	40.31	40.72
2015-04-02 16:00:00	40.66	40.74	40.12	40.29
2015-04-06 16:00:00	40.34	41.78	40.18	41.55
2015-04-07 16:00:00	41.61	41.91	41.31	41.53
2015-04-08 16:00:00	41.48	41.69	41.04	41.42
...	...	...	...	...
2021-03-25 16:00:00	235.30	236.94	231.57	232.34
2021-03-26 16:00:00	231.55	236.71	231.55	236.48
2021-03-29 16:00:00	236.59	236.80	231.88	235.24
2021-03-30 16:00:00	233.53	233.85	231.10	231.85
2021-03-31 16:00:00	232.91	239.10	232.39	235.77

1511 rows × 4 columns

Checking the null values

```
df.isnull().sum()
```

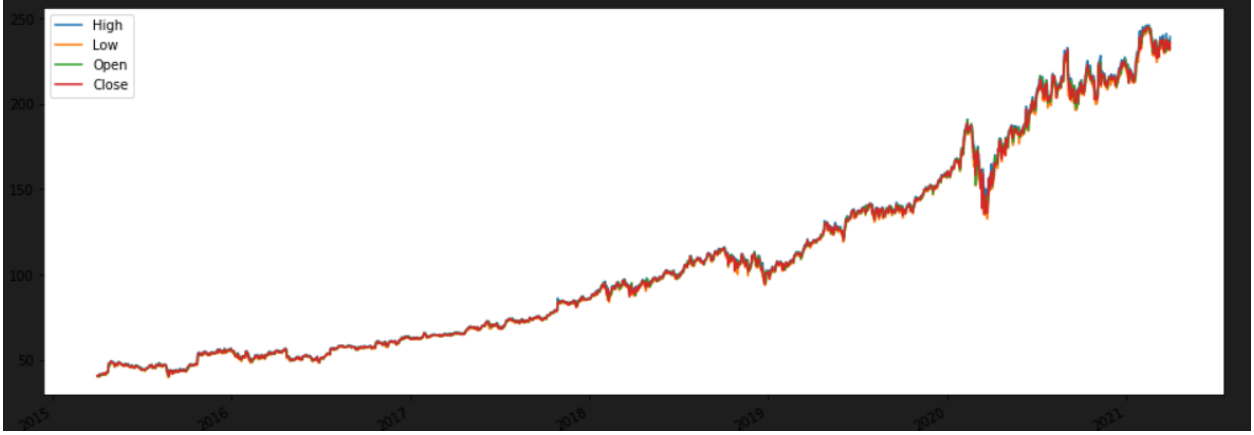
```
Open     0
High     0
Low      0
Close    0
dtype: int64
```

There are no null values so let's move ahead.

Plotting the graph for “ High”, “Low” , “Open”, “Close”.

```
df[['High','Low','Open','Close']].plot(figsize = (16, 6))
```

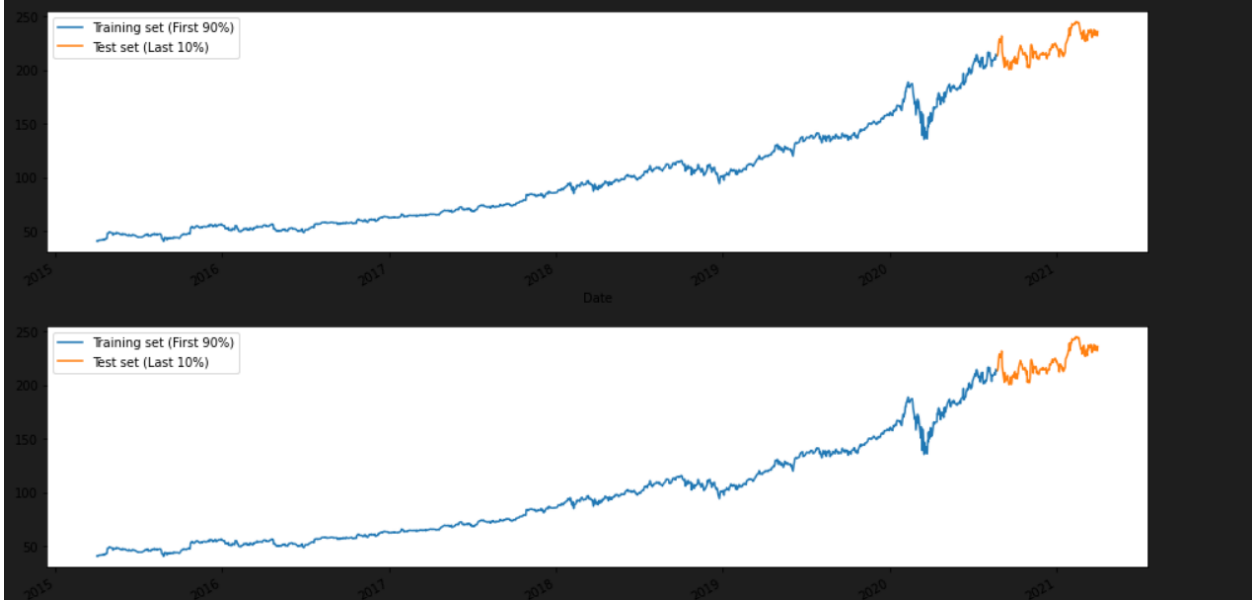
```
<AxesSubplot:xlabel='Date'>
```




```
sc = MinMaxScaler(feature_range=(0,1))

def load_data(datasetname, column, seq_len):
    # A support function to help prepare datasets for an RNN/LSTM/GRU
    data = datasetname.loc[:,column]

    sequence_length = seq_len + 1
    result = []
    for index in range(len(data) - sequence_length):
        result.append(data[index: index + sequence_length])

    result = np.array(result)

    #Last 10% is used for validation test, first 90% for training
    row = round(0.9 * result.shape[0])
    train = result[:int(row), :]
    np.random.shuffle(train)
    x_train = train[:, :-1]
    y_train = train[:, -1]
    x_test = result[int(row):, :-1]
    y_test = result[int(row):, -1]

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

    return [x_train, y_train, x_test, y_test]
```

I split the training and testing data into 90:10 ratio.

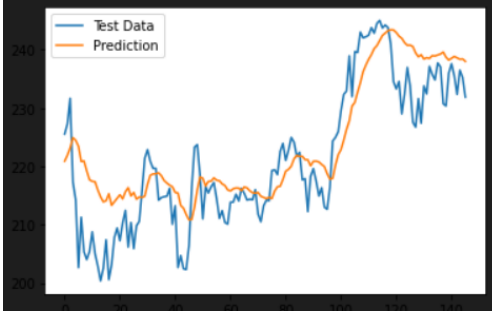
```
X_train, Y_train, X_test, Y_test = load_data(df, 'Close', 50)

df['Close'][:int(len(df['Close']) * .9)].plot(figsize=(16,4),legend=True)
df['Close'][int(len(df['Close']) * .9):].plot(figsize=(16,4),legend=True)
plt.legend(['Training set (First 90%)','Test set (Last 10%)'])
plt.show()
```




```
model=Sequential()

model.add(LSTM(25, activation = 'relu', input_shape = (X_train.shape[1],1)))
model.add(Dense(10))
model.add(Dense(1))

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


model.fit(X_train, Y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
42/42 [==============================] - 3s 28ms/step - loss: 199.1153
Epoch 2/10
42/42 [==============================] - 1s 28ms/step - loss: 29.3240
Epoch 3/10
42/42 [==============================] - 1s 28ms/step - loss: 36.8765
Epoch 4/10
42/42 [==============================] - 1s 30ms/step - loss: 22.0111
Epoch 5/10
42/42 [==============================] - 1s 28ms/step - loss: 12.3427
Epoch 6/10
42/42 [==============================] - 2s 39ms/step - loss: 11.0326
Epoch 7/10
42/42 [==============================] - 1s 28ms/step - loss: 10.3515
Epoch 8/10
42/42 [==============================] - 1s 28ms/step - loss: 10.0735
Epoch 9/10
42/42 [==============================] - 1s 29ms/step - loss: 9.4286
Epoch 10/10
42/42 [==============================] - 1s 28ms/step - loss: 12.2310

<keras.callbacks.History at 0x7f4a7c05fc90>
```

```python
prediction = model.predict(X_test)


plt.plot(Y_test, label='Test Data')
plt.plot(prediction, label='Prediction')
plt.legend()
plt.show()
```



Conclusion:

1. In Time series analysis a feature is compared over a time interval to see its dependency on time.
2. Stock market is one of the most important domains for time series analysis.
3. So I have Performed time series analysis on Microsoft stock.
4. I have used the LSTM (Long Short Term Memory) Algorithm for sequential prediction.