IBM NAAN MUDHALVAN

PHASE-4 PROJECT SUBMISSION

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

program:

1.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
```

2.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
```

3.Describe the dataset:

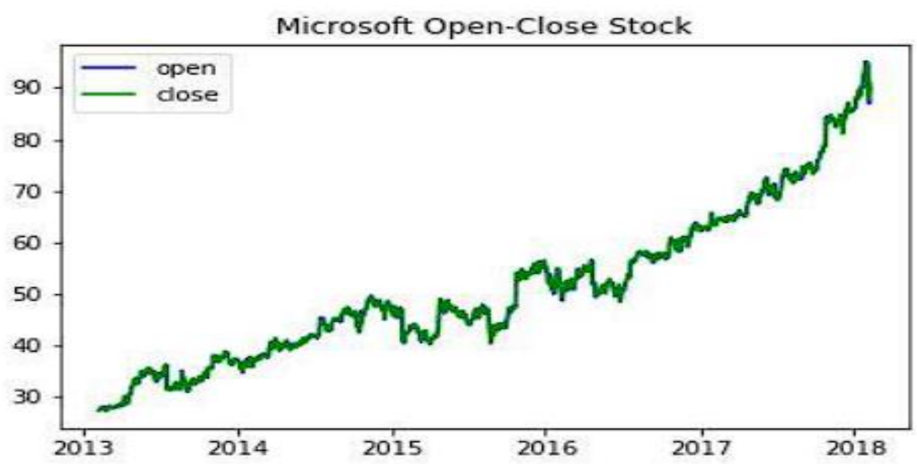
```
df.describe()
```

	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

4.Data Visualization:

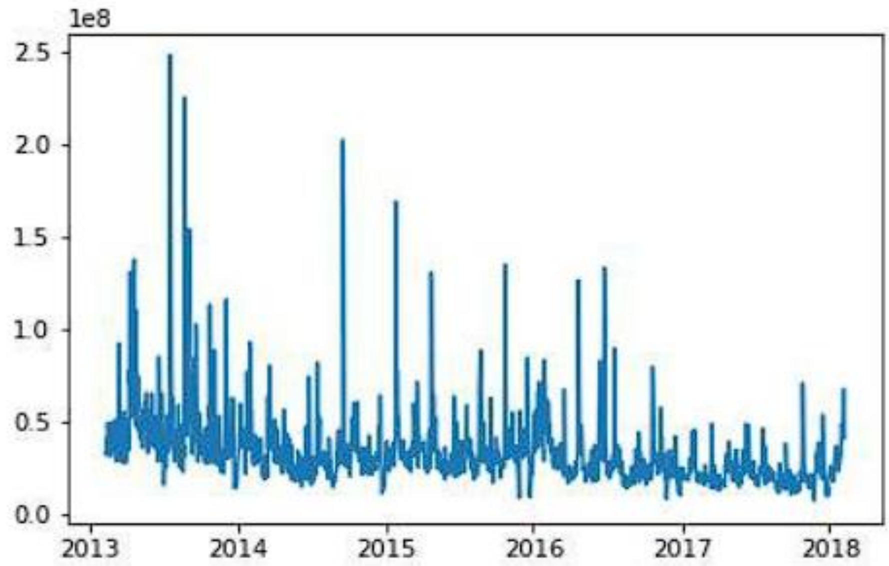
```
plt.plot(df['date'],
         df['open'],
         color="blue",
         label="open")
plt.plot(df['date'],
         df['close'],
         color="green",
         label="close")
plt.title("Microsoft Open-Close Stock")
plt.legend()
```

Output



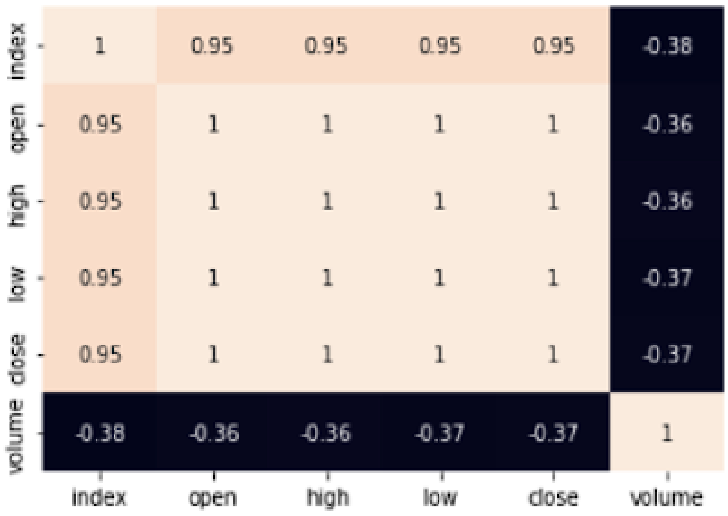
```python
plt.plot(df['date'],
         df['volume'])
plt.show()
```

Output:




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

Output:



5.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))
```

6.Build the model:

```
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
```

7.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)
```

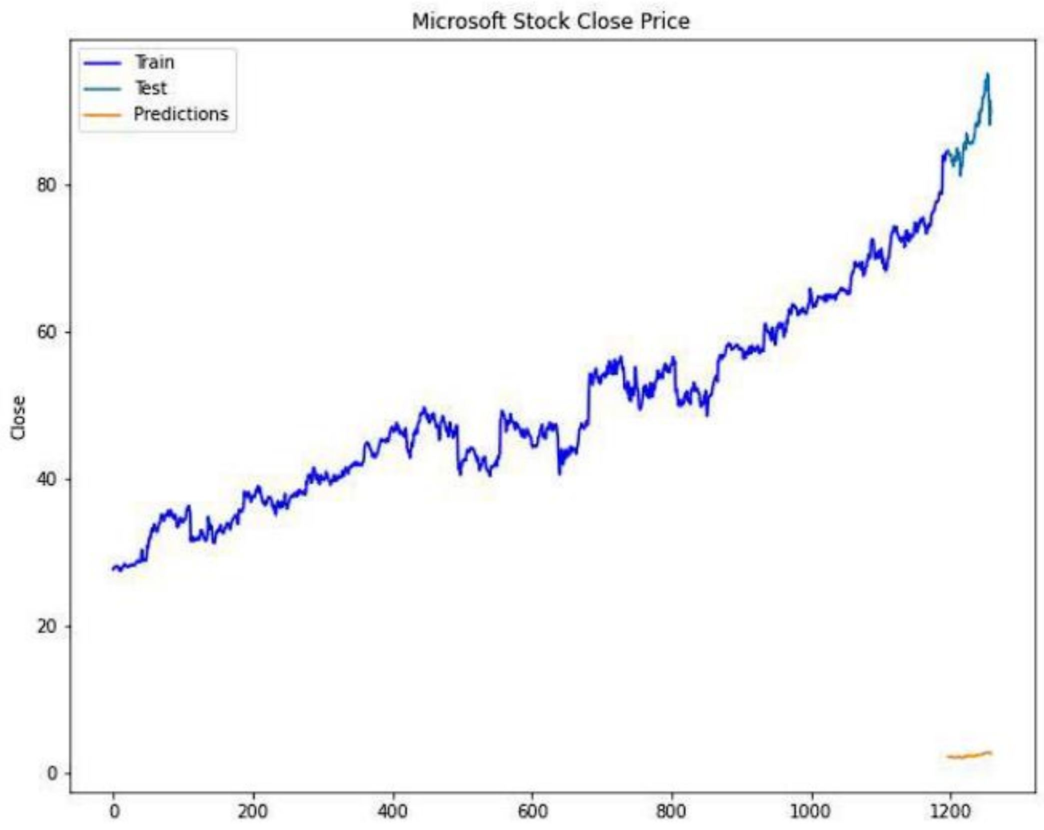
Output:

```
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:



Microsoft Stock Close Price
Train
Test
Predictions
Close
0
20
40
60
80
0
200
400
600
800
1000
1200