```
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
```

```
microsoft = pd.read_csv('/content/MicrosoftStock.csv')
print(microsoft.head())
```

```
   index        date   open   high    low  close    volume  Name
0  390198  2013-02-08  27.35  27.71  27.31  27.55  33318306  MSFT
1  390199  2013-02-11  27.65  27.92  27.50  27.86  32247549  MSFT
2  390200  2013-02-12  27.88  28.00  27.75  27.88  35990829  MSFT
3  390201  2013-02-13  27.93  28.11  27.88  28.03  41715530  MSFT
4  390202  2013-02-14  27.92  28.06  27.87  28.04  32663174  MSFT
```

```
microsoft.shape
```

```
(1259, 8)
```

```
microsoft.info()
```

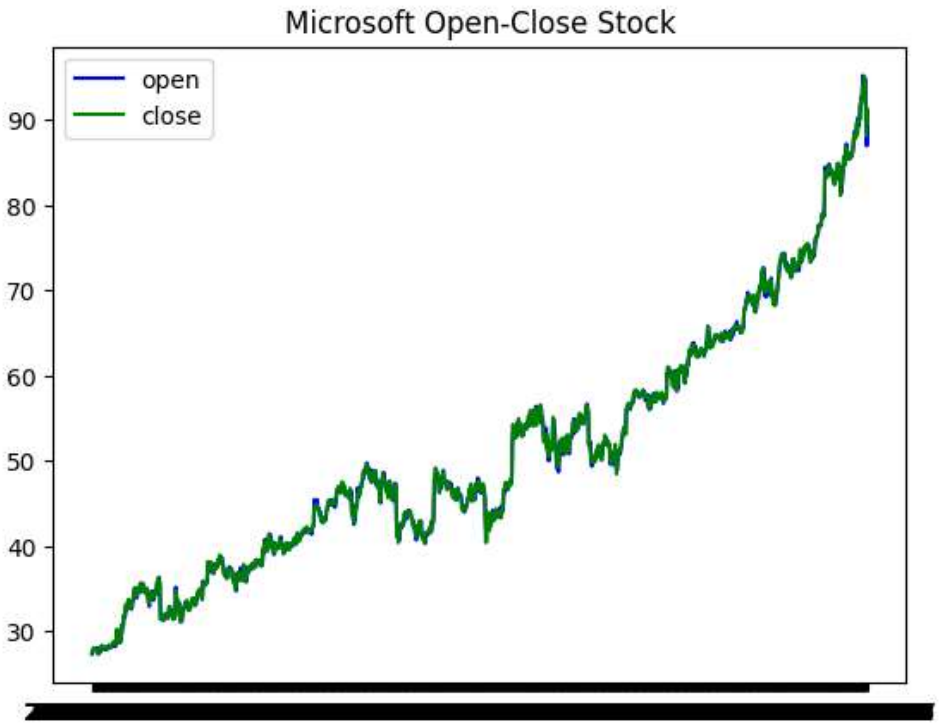
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1259 entries, 0 to 1258
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   index   1259 non-null   int64  
 1   date    1259 non-null   object 
 2   open    1259 non-null   float64
 3   high    1259 non-null   float64
 4   low     1259 non-null   float64
 5   close   1259 non-null   float64
 6   volume  1259 non-null   int64  
 7   Name    1259 non-null   object 
dtypes: float64(4), int64(2), object(2)
memory usage: 78.8+ KB
```

```
microsoft.describe()
```

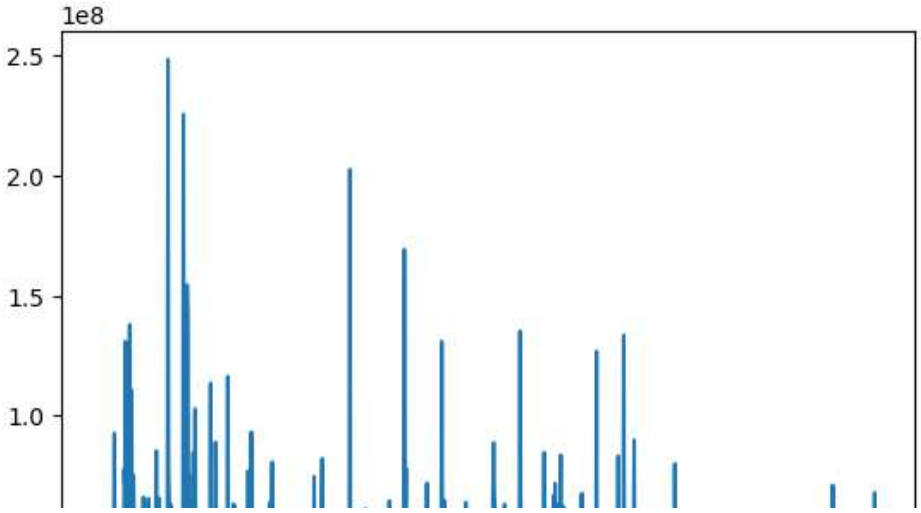
	index	open	high	low	close	volume
count	1259.000000	1259.000000	1259.000000	1259.000000	1259.000000	1.259000e+03
mean	390827.000000	51.026394	51.436007	50.630397	51.063081	3.386946e+07
std	363.586303	14.859387	14.930144	14.774630	14.852117	1.958979e+07
min	390198.000000	27.350000	27.600000	27.230000	27.370000	7.425603e+06

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

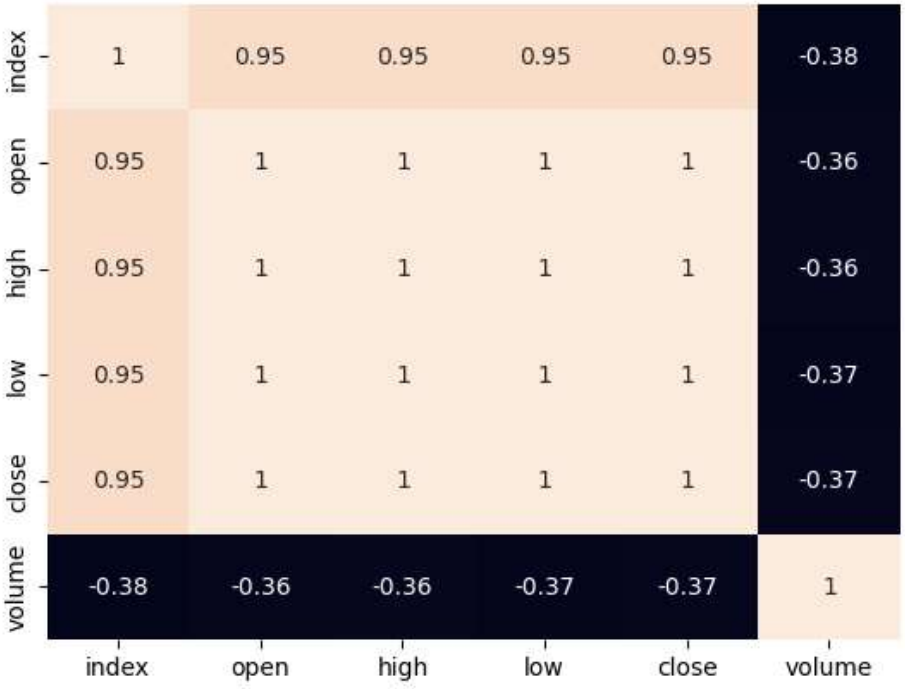
```
<matplotlib.legend.Legend at 0x7b0560566650>
```




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



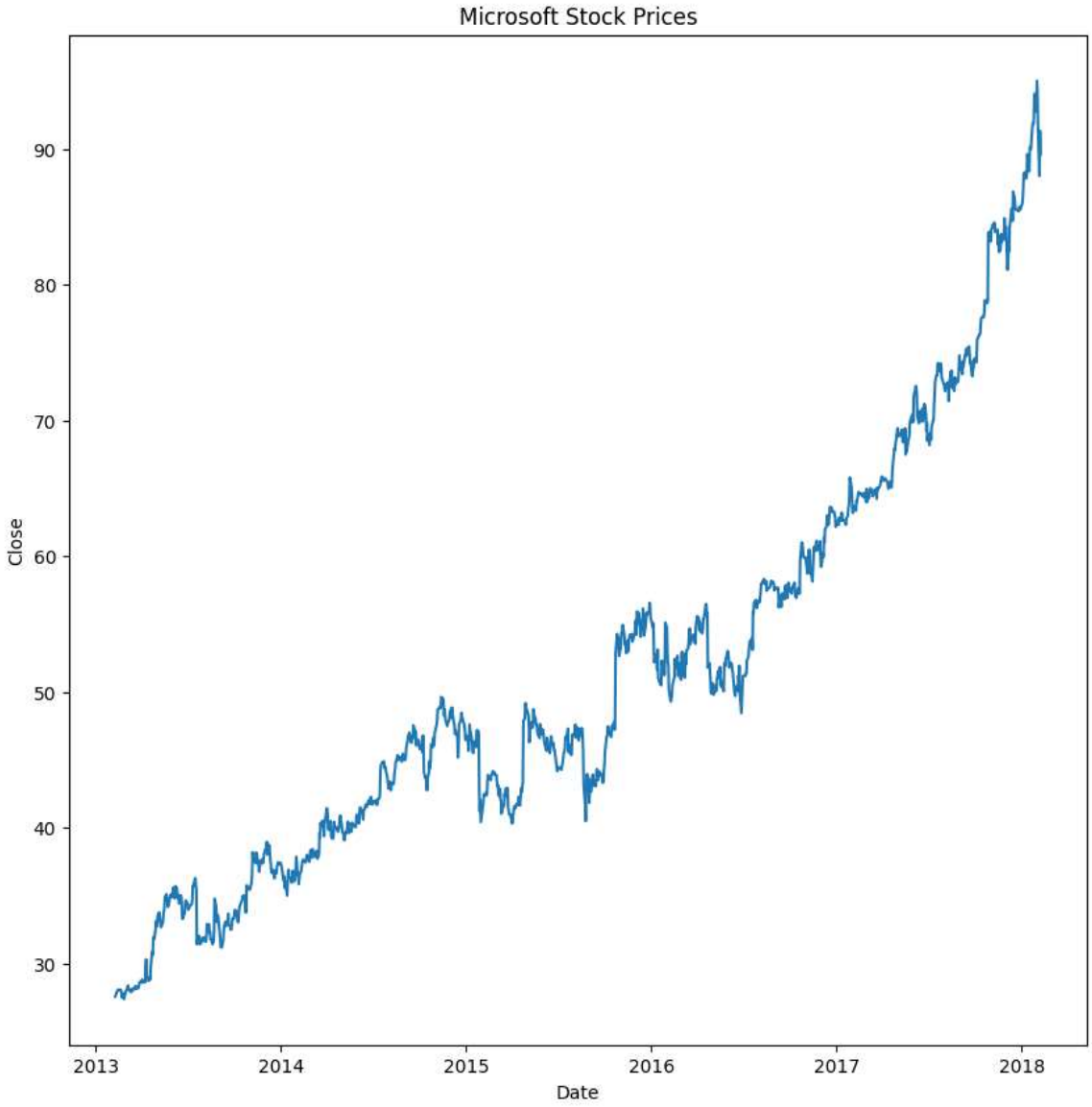

```
sns.heatmap(microsoft.corr(),
            annot=True,
            cbar=False)
plt.show()
```




```
microsoft['date'] = pd.to_datetime(microsoft['date'])
prediction = microsoft.loc[(microsoft['date']
              > datetime(2013, 1, 1))
              & (microsoft['date']
              < datetime(2018, 1, 1))]

plt.figure(figsize=(10, 10))
plt.plot(microsoft['date'], microsoft['close'])
plt.xlabel("Date")
plt.ylabel("Close")
plt.title("Microsoft Stock Prices")
```

```
Text(0.5, 1.0, 'Microsoft Stock Prices')
```




```
# prepare the training set samples
msft_close = microsoft.filter(['close'])
dataset = msft_close.values
# Assuming 'dataset' is the variable that contains your dataset
# Calculate the training set size as 95% of the total dataset size
training = int(np.ceil(len(dataset) * 0.95))
```

```
# scale the data
ss = StandardScaler()
ss = ss.fit_transform(dataset)
```

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

x_train = []
y_train = []

# considering 60 as the batch size,
# create the X_train and 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())
```

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

 lstm_1 (LSTM)               (None, 64)                33024

 dense_3 (Dense)             (None, 128)               8320

 dropout_1 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 1)                 129

=================================================================
Total params: 58369 (228.00 KB)
Trainable params: 58369 (228.00 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

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

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

```
Epoch 1/20
36/36 [==============================] - 4s 39ms/step - loss: 0.2009 - root_mean_squared_error: 0.3008
Epoch 2/20
36/36 [==============================] - 1s 38ms/step - loss: 0.1196 - root_mean_squared_error: 0.1624
Epoch 3/20
36/36 [==============================] - 2s 49ms/step - loss: 0.1076 - root_mean_squared_error: 0.1437
Epoch 4/20
36/36 [==============================] - 2s 44ms/step - loss: 0.0957 - root_mean_squared_error: 0.1300
Epoch 5/20
36/36 [==============================] - 1s 38ms/step - loss: 0.0878 - root_mean_squared_error: 0.1176
Epoch 6/20
36/36 [==============================] - 1s 38ms/step - loss: 0.0869 - root_mean_squared_error: 0.1174
Epoch 7/20
36/36 [==============================] - 1s 39ms/step - loss: 0.0868 - root_mean_squared_error: 0.1173
Epoch 8/20
36/36 [==============================] - 1s 39ms/step - loss: 0.0805 - root_mean_squared_error: 0.1076
Epoch 9/20
36/36 [==============================] - 1s 39ms/step - loss: 0.0879 - root_mean_squared_error: 0.1187
Epoch 10/20
36/36 [==============================] - 1s 38ms/step - loss: 0.0832 - root_mean_squared_error: 0.1144
Epoch 11/20
36/36 [==============================] - 2s 47ms/step - loss: 0.0866 - root_mean_squared_error: 0.1121
Epoch 12/20
36/36 [==============================] - 2s 44ms/step - loss: 0.0833 - root_mean_squared_error: 0.1092
Epoch 13/20
36/36 [==============================] - 1s 37ms/step - loss: 0.0883 - root_mean_squared_error: 0.1176
Epoch 14/20
36/36 [==============================] - 1s 39ms/step - loss: 0.0802 - root_mean_squared_error: 0.1067
Epoch 15/20
36/36 [==============================] - 1s 38ms/step - loss: 0.0822 - root_mean_squared_error: 0.1114
Epoch 16/20
36/36 [==============================] - 1s 39ms/step - loss: 0.0780 - root_mean_squared_error: 0.1048
Epoch 17/20
36/36 [==============================] - 1s 38ms/step - loss: 0.0831 - root_mean_squared_error: 0.1091
Epoch 18/20
36/36 [==============================] - 1s 39ms/step - loss: 0.0838 - root_mean_squared_error: 0.1137
Epoch 19/20
36/36 [==============================] - 2s 48ms/step - loss: 0.0755 - root_mean_squared_error: 0.1015
Epoch 20/20
36/36 [==============================] - 2s 46ms/step - loss: 0.0783 - root_mean_squared_error: 0.1068
```

```
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)
```

```
2/2 [==============================] - 1s 15ms/step
```

```
train = microsoft[:training]
test = microsoft[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'])
```

```
<matplotlib.legend.Legend at 0x7b05683ed240>
```