n37xougvi

September 11, 2023

[1]:
```
#importing libraries to be used
import numpy as np # for linear algebra
import pandas as pd # data preprocessing
import matplotlib.pyplot as plt # data visualization library
import seaborn as sns # data visualization library
%matplotlib inline
import warnings
warnings.filterwarnings('ignore') # ignore warnings

from sklearn.preprocessing import MinMaxScaler # for normalization
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, Bidirectional
```

[3]:
```
df = pd.read_csv('Task_1_Stocks_dataset.csv') # data_importing
df.head(10) # fetching first 10 rows of dataset
```

[3]:
```
  symbol                       date   close    high       low    open  \
0   GOOG  2016-06-14 00:00:00+00:00  718.27  722.47  713.1200  716.48
1   GOOG  2016-06-15 00:00:00+00:00  718.92  722.98  717.3100  719.00
2   GOOG  2016-06-16 00:00:00+00:00  710.36  716.65  703.2600  714.91
3   GOOG  2016-06-17 00:00:00+00:00  691.72  708.82  688.4515  708.65
4   GOOG  2016-06-20 00:00:00+00:00  693.71  702.48  693.4100  698.77
5   GOOG  2016-06-21 00:00:00+00:00  695.94  702.77  692.0100  698.40
6   GOOG  2016-06-22 00:00:00+00:00  697.46  700.86  693.0819  699.06
7   GOOG  2016-06-23 00:00:00+00:00  701.87  701.95  687.0000  697.45
8   GOOG  2016-06-24 00:00:00+00:00  675.22  689.40  673.4500  675.17
9   GOOG  2016-06-27 00:00:00+00:00  668.26  672.30  663.2840  671.00

    volume  adjClose  adjHigh    adjLow  adjOpen  adjVolume  divCash  \
0  1306065    718.27   722.47  713.1200   716.48    1306065      0.0
1  1214517    718.92   722.98  717.3100   719.00    1214517      0.0
2  1982471    710.36   716.65  703.2600   714.91    1982471      0.0
3  3402357    691.72   708.82  688.4515   708.65    3402357      0.0
4  2082538    693.71   702.48  693.4100   698.77    2082538      0.0
5  1465634    695.94   702.77  692.0100   698.40    1465634      0.0
6  1184318    697.46   700.86  693.0819   699.06    1184318      0.0
7  2171415    701.87   701.95  687.0000   697.45    2171415      0.0
```

```
8  4449022    675.22   689.40  673.4500   675.17    4449022      0.0
9  2641085    668.26   672.30  663.2840   671.00    2641085      0.0

   splitFactor
0          1.0
1          1.0
2          1.0
3          1.0
4          1.0
5          1.0
6          1.0
7          1.0
8          1.0
9          1.0
```

[4]:
```python
# shape of data
print("Shape of data:",df.shape)
```

```
Shape of data: (1258, 14)
```

[5]:
```python
# statistical description of data
df.describe()
```

[5]:
```
             close         high          low         open        volume  \
count  1258.000000  1258.000000  1258.000000  1258.000000  1.258000e+03
mean   1216.317067  1227.430934  1204.176430  1215.260779  1.601590e+06
std     383.333358   387.570872   378.777094   382.446995  6.960172e+05
min     668.260000   672.300000   663.284000   671.000000  3.467530e+05
25%     960.802500   968.757500   952.182500   959.005000  1.173522e+06
50%    1132.460000  1143.935000  1117.915000  1131.150000  1.412588e+06
75%    1360.595000  1374.345000  1348.557500  1361.075000  1.812156e+06
max    2521.600000  2526.990000  2498.290000  2524.920000  6.207027e+06

          adjClose      adjHigh       adjLow      adjOpen     adjVolume  \
count  1258.000000  1258.000000  1258.000000  1258.000000  1.258000e+03
mean   1216.317067  1227.430936  1204.176436  1215.260779  1.601590e+06
std     383.333358   387.570873   378.777099   382.446995  6.960172e+05
min     668.260000   672.300000   663.284000   671.000000  3.467530e+05
25%     960.802500   968.757500   952.182500   959.005000  1.173522e+06
50%    1132.460000  1143.935000  1117.915000  1131.150000  1.412588e+06
75%    1360.595000  1374.345000  1348.557500  1361.075000  1.812156e+06
max    2521.600000  2526.990000  2498.290000  2524.920000  6.207027e+06

       divCash  splitFactor
count   1258.0       1258.0
mean       0.0          1.0
std        0.0          0.0
```

```
min        0.0         1.0
25%        0.0         1.0
50%        0.0         1.0
75%        0.0         1.0
max        0.0         1.0
```

[6]:
```
# summary of data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   symbol       1258 non-null   object
 1   date         1258 non-null   object
 2   close        1258 non-null   float64
 3   high         1258 non-null   float64
 4   low          1258 non-null   float64
 5   open         1258 non-null   float64
 6   volume       1258 non-null   int64
 7   adjClose     1258 non-null   float64
 8   adjHigh      1258 non-null   float64
 9   adjLow       1258 non-null   float64
 10  adjOpen      1258 non-null   float64
 11  adjVolume    1258 non-null   int64
 12  divCash      1258 non-null   float64
 13  splitFactor  1258 non-null   float64
dtypes: float64(10), int64(2), object(2)
memory usage: 137.7+ KB
```

[7]:
```
# checking null values
df.isnull().sum()
```

[7]:
```
symbol         0
date           0
close          0
high           0
low            0
open           0
volume         0
adjClose       0
adjHigh        0
adjLow         0
adjOpen        0
adjVolume      0
divCash        0
```

```
splitFactor    0
dtype: int64
```

[8]:
```
df = df[['date','open','close']] # Extracting required columns
df['date'] = pd.to_datetime(df['date'].apply(lambda x: x.split()[0])) #␣
↪converting object dtype of date column to datetime dtype
df.set_index('date',drop=True,inplace=True) # Setting date column as index
df.head(10)
```

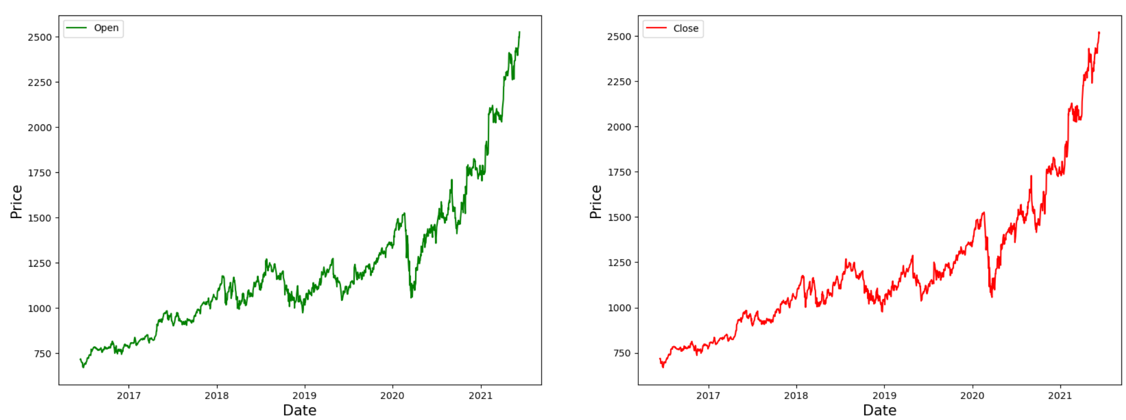
[8]:

date	open	close
2016-06-14	716.48	718.27
2016-06-15	719.00	718.92
2016-06-16	714.91	710.36
2016-06-17	708.65	691.72
2016-06-20	698.77	693.71
2016-06-21	698.40	695.94
2016-06-22	699.06	697.46
2016-06-23	697.45	701.87
2016-06-24	675.17	675.22
2016-06-27	671.00	668.26

[9]:
```
# plotting open and closing price on date index
fig, ax =plt.subplots(1,2,figsize=(20,7))
ax[0].plot(df['open'],label='Open',color='green')
ax[0].set_xlabel('Date',size=15)
ax[0].set_ylabel('Price',size=15)
ax[0].legend()

ax[1].plot(df['close'],label='Close',color='red')
ax[1].set_xlabel('Date',size=15)
ax[1].set_ylabel('Price',size=15)
ax[1].legend()

fig.show()
```



[10]:
```python
# normalizing all the values of all columns using MinMaxScaler
MMS = MinMaxScaler()
df[df.columns] = MMS.fit_transform(df)
df.head(10)
```

[10]:
```
                open     close
date
2016-06-14  0.024532  0.026984
2016-06-15  0.025891  0.027334
2016-06-16  0.023685  0.022716
2016-06-17  0.020308  0.012658
2016-06-20  0.014979  0.013732
2016-06-21  0.014779  0.014935
2016-06-22  0.015135  0.015755
2016-06-23  0.014267  0.018135
2016-06-24  0.002249  0.003755
2016-06-27  0.000000  0.000000
```

[11]:
```python
# splitting the data into training and test set
training_size = round(len(df) * 0.75) # Selecting 75 % for training and 25 %␣
↪for testing
training_size
```

[11]: 944

[12]:
```python
train_data = df[:training_size]
test_data  = df[training_size:]

train_data.shape, test_data.shape
```

[12]: ((944, 2), (314, 2))

[13]:
```python
# Function to create sequence of data for training and testing

def create_sequence(dataset):
  sequences = []
  labels = []

  start_idx = 0

  for stop_idx in range(50,len(dataset)): # Selecting 50 rows at a time
    sequences.append(dataset.iloc[start_idx:stop_idx])
    labels.append(dataset.iloc[stop_idx])
    start_idx += 1
  return (np.array(sequences),np.array(labels))
```

[14]:
```
train_seq, train_label = create_sequence(train_data)
test_seq, test_label = create_sequence(test_data)
train_seq.shape, train_label.shape, test_seq.shape, test_label.shape
```

[14]: ((894, 50, 2), (894, 2), (264, 50, 2), (264, 2))

[15]:
```
# imported Sequential from keras.models
model = Sequential()
# importing Dense, Dropout, LSTM, Bidirectional from keras.layers
model.add(LSTM(units=50, return_sequences=True, input_shape = (train_seq.
 ↪shape[1], train_seq.shape[2])))

model.add(Dropout(0.1))
model.add(LSTM(units=50))

model.add(Dense(2))

model.compile(loss='mean_squared_error', optimizer='adam',␣
 ↪metrics=['mean_absolute_error'])

model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 50, 50)            10600

 dropout (Dropout)           (None, 50, 50)            0

 lstm_1 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 2)                 102

=================================================================
Total params: 30902 (120.71 KB)
Trainable params: 30902 (120.71 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[16]:
```
# fitting the model by iterating the dataset over 100 times(100 epochs)
model.fit(train_seq, train_label, epochs=100,validation_data=(test_seq,␣
 ↪test_label), verbose=1)
```

```
Epoch 1/100
28/28 [==============================] - 8s 96ms/step - loss: 0.0097 -
mean_absolute_error: 0.0700 - val_loss: 0.0211 - val_mean_absolute_error: 0.1204
```

```
Epoch 2/100
28/28 [==============================] - 2s 56ms/step - loss: 0.0011 -
mean_absolute_error: 0.0267 - val_loss: 0.0072 - val_mean_absolute_error: 0.0670
Epoch 3/100
28/28 [==============================] - 1s 52ms/step - loss: 5.1076e-04 -
mean_absolute_error: 0.0170 - val_loss: 0.0032 - val_mean_absolute_error: 0.0433
Epoch 4/100
28/28 [==============================] - 1s 44ms/step - loss: 4.9178e-04 -
mean_absolute_error: 0.0161 - val_loss: 0.0058 - val_mean_absolute_error: 0.0604
Epoch 5/100
28/28 [==============================] - 1s 44ms/step - loss: 4.3928e-04 -
mean_absolute_error: 0.0152 - val_loss: 0.0059 - val_mean_absolute_error: 0.0611
Epoch 6/100
28/28 [==============================] - 1s 43ms/step - loss: 4.4351e-04 -
mean_absolute_error: 0.0154 - val_loss: 0.0033 - val_mean_absolute_error: 0.0433
Epoch 7/100
28/28 [==============================] - 1s 42ms/step - loss: 4.5734e-04 -
mean_absolute_error: 0.0155 - val_loss: 0.0043 - val_mean_absolute_error: 0.0502
Epoch 8/100
28/28 [==============================] - 1s 43ms/step - loss: 4.2159e-04 -
mean_absolute_error: 0.0150 - val_loss: 0.0062 - val_mean_absolute_error: 0.0629
Epoch 9/100
28/28 [==============================] - 1s 42ms/step - loss: 3.9659e-04 -
mean_absolute_error: 0.0146 - val_loss: 0.0047 - val_mean_absolute_error: 0.0531
Epoch 10/100
28/28 [==============================] - 1s 43ms/step - loss: 4.1426e-04 -
mean_absolute_error: 0.0148 - val_loss: 0.0043 - val_mean_absolute_error: 0.0503
Epoch 11/100
28/28 [==============================] - 1s 43ms/step - loss: 3.8902e-04 -
mean_absolute_error: 0.0142 - val_loss: 0.0057 - val_mean_absolute_error: 0.0597
Epoch 12/100
28/28 [==============================] - 1s 44ms/step - loss: 3.9106e-04 -
mean_absolute_error: 0.0145 - val_loss: 0.0057 - val_mean_absolute_error: 0.0599
Epoch 13/100
28/28 [==============================] - 1s 45ms/step - loss: 4.0174e-04 -
mean_absolute_error: 0.0147 - val_loss: 0.0047 - val_mean_absolute_error: 0.0529
Epoch 14/100
28/28 [==============================] - 1s 43ms/step - loss: 3.6485e-04 -
mean_absolute_error: 0.0138 - val_loss: 0.0044 - val_mean_absolute_error: 0.0506
Epoch 15/100
28/28 [==============================] - 1s 43ms/step - loss: 3.5733e-04 -
mean_absolute_error: 0.0138 - val_loss: 0.0057 - val_mean_absolute_error: 0.0589
Epoch 16/100
28/28 [==============================] - 1s 44ms/step - loss: 3.3488e-04 -
mean_absolute_error: 0.0135 - val_loss: 0.0073 - val_mean_absolute_error: 0.0692
Epoch 17/100
28/28 [==============================] - 1s 43ms/step - loss: 3.2153e-04 -
mean_absolute_error: 0.0130 - val_loss: 0.0068 - val_mean_absolute_error: 0.0659
```

```
Epoch 18/100
28/28 [==============================] - 1s 42ms/step - loss: 3.3644e-04 -
mean_absolute_error: 0.0135 - val_loss: 0.0096 - val_mean_absolute_error: 0.0802
Epoch 19/100
28/28 [==============================] - 1s 43ms/step - loss: 3.9048e-04 -
mean_absolute_error: 0.0148 - val_loss: 0.0036 - val_mean_absolute_error: 0.0445
Epoch 20/100
28/28 [==============================] - 1s 43ms/step - loss: 3.6870e-04 -
mean_absolute_error: 0.0140 - val_loss: 0.0080 - val_mean_absolute_error: 0.0733
Epoch 21/100
28/28 [==============================] - 1s 43ms/step - loss: 3.3977e-04 -
mean_absolute_error: 0.0134 - val_loss: 0.0037 - val_mean_absolute_error: 0.0452
Epoch 22/100
28/28 [==============================] - 1s 44ms/step - loss: 3.1535e-04 -
mean_absolute_error: 0.0130 - val_loss: 0.0053 - val_mean_absolute_error: 0.0561
Epoch 23/100
28/28 [==============================] - 1s 43ms/step - loss: 2.9157e-04 -
mean_absolute_error: 0.0126 - val_loss: 0.0062 - val_mean_absolute_error: 0.0620
Epoch 24/100
28/28 [==============================] - 1s 42ms/step - loss: 3.1603e-04 -
mean_absolute_error: 0.0130 - val_loss: 0.0066 - val_mean_absolute_error: 0.0632
Epoch 25/100
28/28 [==============================] - 1s 42ms/step - loss: 2.8856e-04 -
mean_absolute_error: 0.0125 - val_loss: 0.0074 - val_mean_absolute_error: 0.0682
Epoch 26/100
28/28 [==============================] - 1s 43ms/step - loss: 2.8494e-04 -
mean_absolute_error: 0.0125 - val_loss: 0.0065 - val_mean_absolute_error: 0.0645
Epoch 27/100
28/28 [==============================] - 1s 43ms/step - loss: 2.9341e-04 -
mean_absolute_error: 0.0126 - val_loss: 0.0061 - val_mean_absolute_error: 0.0605
Epoch 28/100
28/28 [==============================] - 1s 43ms/step - loss: 2.7762e-04 -
mean_absolute_error: 0.0121 - val_loss: 0.0055 - val_mean_absolute_error: 0.0561
Epoch 29/100
28/28 [==============================] - 1s 43ms/step - loss: 2.8144e-04 -
mean_absolute_error: 0.0124 - val_loss: 0.0039 - val_mean_absolute_error: 0.0453
Epoch 30/100
28/28 [==============================] - 1s 42ms/step - loss: 2.6566e-04 -
mean_absolute_error: 0.0119 - val_loss: 0.0049 - val_mean_absolute_error: 0.0529
Epoch 31/100
28/28 [==============================] - 1s 43ms/step - loss: 2.5952e-04 -
mean_absolute_error: 0.0118 - val_loss: 0.0063 - val_mean_absolute_error: 0.0614
Epoch 32/100
28/28 [==============================] - 1s 43ms/step - loss: 2.8308e-04 -
mean_absolute_error: 0.0124 - val_loss: 0.0055 - val_mean_absolute_error: 0.0568
Epoch 33/100
28/28 [==============================] - 1s 43ms/step - loss: 2.6845e-04 -
mean_absolute_error: 0.0121 - val_loss: 0.0049 - val_mean_absolute_error: 0.0529
```

```
Epoch 34/100
28/28 [==============================] - 1s 43ms/step - loss: 2.4217e-04 -
mean_absolute_error: 0.0114 - val_loss: 0.0063 - val_mean_absolute_error: 0.0618
Epoch 35/100
28/28 [==============================] - 1s 43ms/step - loss: 2.4681e-04 -
mean_absolute_error: 0.0116 - val_loss: 0.0105 - val_mean_absolute_error: 0.0817
Epoch 36/100
28/28 [==============================] - 1s 43ms/step - loss: 2.4919e-04 -
mean_absolute_error: 0.0115 - val_loss: 0.0053 - val_mean_absolute_error: 0.0546
Epoch 37/100
28/28 [==============================] - 1s 44ms/step - loss: 2.5193e-04 -
mean_absolute_error: 0.0118 - val_loss: 0.0038 - val_mean_absolute_error: 0.0440
Epoch 38/100
28/28 [==============================] - 1s 43ms/step - loss: 2.5658e-04 -
mean_absolute_error: 0.0117 - val_loss: 0.0047 - val_mean_absolute_error: 0.0521
Epoch 39/100
28/28 [==============================] - 1s 44ms/step - loss: 2.3495e-04 -
mean_absolute_error: 0.0112 - val_loss: 0.0037 - val_mean_absolute_error: 0.0444
Epoch 40/100
28/28 [==============================] - 1s 42ms/step - loss: 2.6157e-04 -
mean_absolute_error: 0.0116 - val_loss: 0.0066 - val_mean_absolute_error: 0.0619
Epoch 41/100
28/28 [==============================] - 1s 43ms/step - loss: 2.1096e-04 -
mean_absolute_error: 0.0107 - val_loss: 0.0090 - val_mean_absolute_error: 0.0774
Epoch 42/100
28/28 [==============================] - 1s 43ms/step - loss: 2.3469e-04 -
mean_absolute_error: 0.0111 - val_loss: 0.0061 - val_mean_absolute_error: 0.0593
Epoch 43/100
28/28 [==============================] - 1s 42ms/step - loss: 2.1335e-04 -
mean_absolute_error: 0.0106 - val_loss: 0.0057 - val_mean_absolute_error: 0.0587
Epoch 44/100
28/28 [==============================] - 1s 43ms/step - loss: 2.1056e-04 -
mean_absolute_error: 0.0107 - val_loss: 0.0053 - val_mean_absolute_error: 0.0551
Epoch 45/100
28/28 [==============================] - 1s 42ms/step - loss: 2.1404e-04 -
mean_absolute_error: 0.0108 - val_loss: 0.0039 - val_mean_absolute_error: 0.0453
Epoch 46/100
28/28 [==============================] - 1s 42ms/step - loss: 2.1419e-04 -
mean_absolute_error: 0.0108 - val_loss: 0.0033 - val_mean_absolute_error: 0.0413
Epoch 47/100
28/28 [==============================] - 1s 44ms/step - loss: 2.3259e-04 -
mean_absolute_error: 0.0111 - val_loss: 0.0043 - val_mean_absolute_error: 0.0489
Epoch 48/100
28/28 [==============================] - 1s 43ms/step - loss: 1.9071e-04 -
mean_absolute_error: 0.0103 - val_loss: 0.0036 - val_mean_absolute_error: 0.0440
Epoch 49/100
28/28 [==============================] - 1s 43ms/step - loss: 2.0769e-04 -
mean_absolute_error: 0.0105 - val_loss: 0.0037 - val_mean_absolute_error: 0.0448
```

```
Epoch 50/100
28/28 [==============================] - 1s 44ms/step - loss: 1.8372e-04 -
mean_absolute_error: 0.0098 - val_loss: 0.0037 - val_mean_absolute_error: 0.0457
Epoch 51/100
28/28 [==============================] - 1s 42ms/step - loss: 1.8974e-04 -
mean_absolute_error: 0.0100 - val_loss: 0.0031 - val_mean_absolute_error: 0.0403
Epoch 52/100
28/28 [==============================] - 1s 42ms/step - loss: 1.8608e-04 -
mean_absolute_error: 0.0098 - val_loss: 0.0042 - val_mean_absolute_error: 0.0494
Epoch 53/100
28/28 [==============================] - 1s 42ms/step - loss: 1.8983e-04 -
mean_absolute_error: 0.0100 - val_loss: 0.0025 - val_mean_absolute_error: 0.0361
Epoch 54/100
28/28 [==============================] - 1s 44ms/step - loss: 1.9558e-04 -
mean_absolute_error: 0.0101 - val_loss: 0.0035 - val_mean_absolute_error: 0.0443
Epoch 55/100
28/28 [==============================] - 1s 43ms/step - loss: 1.7823e-04 -
mean_absolute_error: 0.0097 - val_loss: 0.0023 - val_mean_absolute_error: 0.0344
Epoch 56/100
28/28 [==============================] - 1s 42ms/step - loss: 1.7260e-04 -
mean_absolute_error: 0.0096 - val_loss: 0.0035 - val_mean_absolute_error: 0.0434
Epoch 57/100
28/28 [==============================] - 1s 43ms/step - loss: 1.9677e-04 -
mean_absolute_error: 0.0101 - val_loss: 0.0032 - val_mean_absolute_error: 0.0413
Epoch 58/100
28/28 [==============================] - 1s 44ms/step - loss: 1.7771e-04 -
mean_absolute_error: 0.0097 - val_loss: 0.0039 - val_mean_absolute_error: 0.0472
Epoch 59/100
28/28 [==============================] - 1s 43ms/step - loss: 1.7042e-04 -
mean_absolute_error: 0.0094 - val_loss: 0.0039 - val_mean_absolute_error: 0.0474
Epoch 60/100
28/28 [==============================] - 1s 42ms/step - loss: 1.8309e-04 -
mean_absolute_error: 0.0098 - val_loss: 0.0043 - val_mean_absolute_error: 0.0522
Epoch 61/100
28/28 [==============================] - 1s 43ms/step - loss: 1.7341e-04 -
mean_absolute_error: 0.0096 - val_loss: 0.0048 - val_mean_absolute_error: 0.0521
Epoch 62/100
28/28 [==============================] - 1s 43ms/step - loss: 1.6127e-04 -
mean_absolute_error: 0.0090 - val_loss: 0.0034 - val_mean_absolute_error: 0.0440
Epoch 63/100
28/28 [==============================] - 1s 43ms/step - loss: 1.6055e-04 -
mean_absolute_error: 0.0092 - val_loss: 0.0057 - val_mean_absolute_error: 0.0598
Epoch 64/100
28/28 [==============================] - 1s 43ms/step - loss: 1.6080e-04 -
mean_absolute_error: 0.0092 - val_loss: 0.0040 - val_mean_absolute_error: 0.0468
Epoch 65/100
28/28 [==============================] - 1s 43ms/step - loss: 1.8400e-04 -
mean_absolute_error: 0.0101 - val_loss: 0.0034 - val_mean_absolute_error: 0.0459
```

```
Epoch 66/100
28/28 [==============================] - 1s 43ms/step - loss: 1.9478e-04 -
mean_absolute_error: 0.0101 - val_loss: 0.0031 - val_mean_absolute_error: 0.0405
Epoch 67/100
28/28 [==============================] - 1s 43ms/step - loss: 1.5378e-04 -
mean_absolute_error: 0.0090 - val_loss: 0.0045 - val_mean_absolute_error: 0.0510
Epoch 68/100
28/28 [==============================] - 1s 42ms/step - loss: 1.4510e-04 -
mean_absolute_error: 0.0085 - val_loss: 0.0042 - val_mean_absolute_error: 0.0502
Epoch 69/100
28/28 [==============================] - 1s 43ms/step - loss: 1.4330e-04 -
mean_absolute_error: 0.0086 - val_loss: 0.0023 - val_mean_absolute_error: 0.0339
Epoch 70/100
28/28 [==============================] - 1s 42ms/step - loss: 1.6006e-04 -
mean_absolute_error: 0.0091 - val_loss: 0.0018 - val_mean_absolute_error: 0.0302
Epoch 71/100
28/28 [==============================] - 1s 43ms/step - loss: 1.4526e-04 -
mean_absolute_error: 0.0087 - val_loss: 0.0015 - val_mean_absolute_error: 0.0277
Epoch 72/100
28/28 [==============================] - 1s 43ms/step - loss: 1.4197e-04 -
mean_absolute_error: 0.0084 - val_loss: 0.0029 - val_mean_absolute_error: 0.0406
Epoch 73/100
28/28 [==============================] - 1s 43ms/step - loss: 1.4049e-04 -
mean_absolute_error: 0.0083 - val_loss: 0.0053 - val_mean_absolute_error: 0.0576
Epoch 74/100
28/28 [==============================] - 1s 43ms/step - loss: 1.5844e-04 -
mean_absolute_error: 0.0092 - val_loss: 0.0036 - val_mean_absolute_error: 0.0459
Epoch 75/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3951e-04 -
mean_absolute_error: 0.0085 - val_loss: 0.0030 - val_mean_absolute_error: 0.0397
Epoch 76/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3533e-04 -
mean_absolute_error: 0.0083 - val_loss: 0.0048 - val_mean_absolute_error: 0.0528
Epoch 77/100
28/28 [==============================] - 1s 44ms/step - loss: 1.5949e-04 -
mean_absolute_error: 0.0091 - val_loss: 0.0040 - val_mean_absolute_error: 0.0478
Epoch 78/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3021e-04 -
mean_absolute_error: 0.0082 - val_loss: 0.0040 - val_mean_absolute_error: 0.0484
Epoch 79/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3080e-04 -
mean_absolute_error: 0.0082 - val_loss: 0.0039 - val_mean_absolute_error: 0.0463
Epoch 80/100
28/28 [==============================] - 1s 43ms/step - loss: 1.2223e-04 -
mean_absolute_error: 0.0079 - val_loss: 0.0017 - val_mean_absolute_error: 0.0293
Epoch 81/100
28/28 [==============================] - 1s 44ms/step - loss: 1.2605e-04 -
mean_absolute_error: 0.0081 - val_loss: 0.0033 - val_mean_absolute_error: 0.0428
```

```
Epoch 82/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3586e-04 -
mean_absolute_error: 0.0084 - val_loss: 0.0016 - val_mean_absolute_error: 0.0281
Epoch 83/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3280e-04 -
mean_absolute_error: 0.0083 - val_loss: 0.0035 - val_mean_absolute_error: 0.0443
Epoch 84/100
28/28 [==============================] - 1s 43ms/step - loss: 1.1856e-04 -
mean_absolute_error: 0.0079 - val_loss: 0.0051 - val_mean_absolute_error: 0.0540
Epoch 85/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3250e-04 -
mean_absolute_error: 0.0084 - val_loss: 0.0040 - val_mean_absolute_error: 0.0483
Epoch 86/100
28/28 [==============================] - 1s 43ms/step - loss: 1.1268e-04 -
mean_absolute_error: 0.0075 - val_loss: 0.0046 - val_mean_absolute_error: 0.0499
Epoch 87/100
28/28 [==============================] - 1s 46ms/step - loss: 1.2861e-04 -
mean_absolute_error: 0.0081 - val_loss: 0.0048 - val_mean_absolute_error: 0.0513
Epoch 88/100
28/28 [==============================] - 1s 43ms/step - loss: 1.1422e-04 -
mean_absolute_error: 0.0076 - val_loss: 0.0039 - val_mean_absolute_error: 0.0452
Epoch 89/100
28/28 [==============================] - 1s 42ms/step - loss: 1.2282e-04 -
mean_absolute_error: 0.0081 - val_loss: 0.0022 - val_mean_absolute_error: 0.0339
Epoch 90/100
28/28 [==============================] - 1s 44ms/step - loss: 1.2268e-04 -
mean_absolute_error: 0.0080 - val_loss: 0.0017 - val_mean_absolute_error: 0.0290
Epoch 91/100
28/28 [==============================] - 1s 42ms/step - loss: 1.1070e-04 -
mean_absolute_error: 0.0073 - val_loss: 0.0039 - val_mean_absolute_error: 0.0472
Epoch 92/100
28/28 [==============================] - 1s 42ms/step - loss: 1.2509e-04 -
mean_absolute_error: 0.0081 - val_loss: 0.0036 - val_mean_absolute_error: 0.0440
Epoch 93/100
28/28 [==============================] - 1s 42ms/step - loss: 1.4067e-04 -
mean_absolute_error: 0.0088 - val_loss: 0.0011 - val_mean_absolute_error: 0.0231
Epoch 94/100
28/28 [==============================] - 1s 42ms/step - loss: 1.1581e-04 -
mean_absolute_error: 0.0077 - val_loss: 0.0012 - val_mean_absolute_error: 0.0244
Epoch 95/100
28/28 [==============================] - 1s 43ms/step - loss: 9.9837e-05 -
mean_absolute_error: 0.0071 - val_loss: 8.5083e-04 - val_mean_absolute_error:
0.0208
Epoch 96/100
28/28 [==============================] - 1s 43ms/step - loss: 1.0957e-04 -
mean_absolute_error: 0.0075 - val_loss: 0.0030 - val_mean_absolute_error: 0.0406
Epoch 97/100
28/28 [==============================] - 1s 44ms/step - loss: 1.0305e-04 -
```

```
mean_absolute_error: 0.0073 - val_loss: 0.0022 - val_mean_absolute_error: 0.0330
Epoch 98/100
28/28 [==============================] - 1s 43ms/step - loss: 1.0927e-04 -
mean_absolute_error: 0.0075 - val_loss: 0.0028 - val_mean_absolute_error: 0.0385
Epoch 99/100
28/28 [==============================] - 1s 42ms/step - loss: 1.0911e-04 -
mean_absolute_error: 0.0074 - val_loss: 0.0030 - val_mean_absolute_error: 0.0397
Epoch 100/100
28/28 [==============================] - 1s 43ms/step - loss: 1.0914e-04 -
mean_absolute_error: 0.0074 - val_loss: 0.0024 - val_mean_absolute_error: 0.0362
```

[16]:
```
<keras.src.callbacks.History at 0x23984000580>
```

[17]:
```
# predicting the values after running the model
test_predicted = model.predict(test_seq)
test_predicted[:5]
```

```
9/9 [==============================] - 1s 25ms/step
```

[17]:
```
array([[0.41031292, 0.40676475],
       [0.41080728, 0.4071397 ],
       [0.40892917, 0.40524966],
       [0.41252786, 0.4086474 ],
       [0.4159498 , 0.41190916]], dtype=float32)
```

[18]:
```
# Inversing normalization/scaling on predicted data
test_inverse_predicted = MMS.inverse_transform(test_predicted)
test_inverse_predicted[:5]
```

[18]:
```
array([[1431.6873, 1422.1334],
       [1432.6038, 1422.8282],
       [1429.122 , 1419.3254],
       [1435.7937, 1425.6226],
       [1442.1376, 1431.6677]], dtype=float32)
```

[20]:
```
# Merging actual and predicted data for better visualization
df_merge = pd.concat([df.iloc[-264:].copy(),
                          pd.
 ↪DataFrame(test_inverse_predicted,columns=['open_predicted','close_predicted'],
                                          index=df.iloc[-264:].index)], axis=1)
```

[21]:
```
# Inversing normalization/scaling
df_merge[['open','close']] = MMS.inverse_transform(df_merge[['open','close']])
df_merge.head()
```

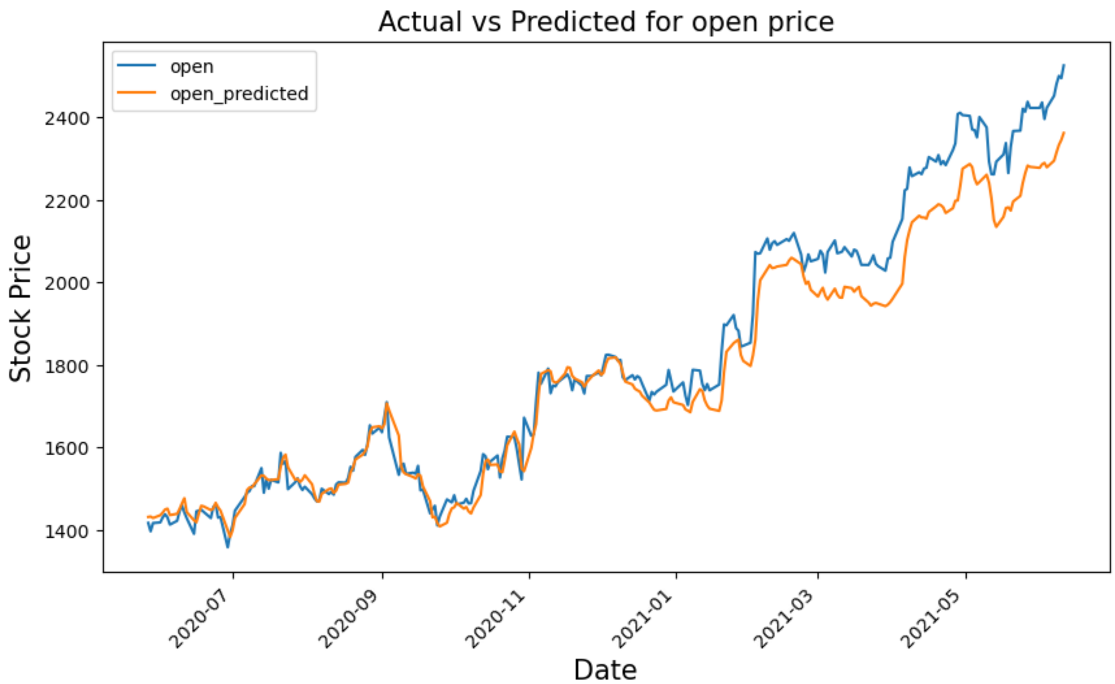
[21]:

	open	close	open_predicted	close_predicted
date				
2020-05-27	1417.25	1417.84	1431.687256	1422.133423

```
2020-05-28  1396.86  1416.73     1432.603760      1422.828247
2020-05-29  1416.94  1428.92     1429.121948      1419.325439
2020-06-01  1418.39  1431.82     1435.793701      1425.622559
2020-06-02  1430.55  1439.22     1442.137573      1431.667725
```

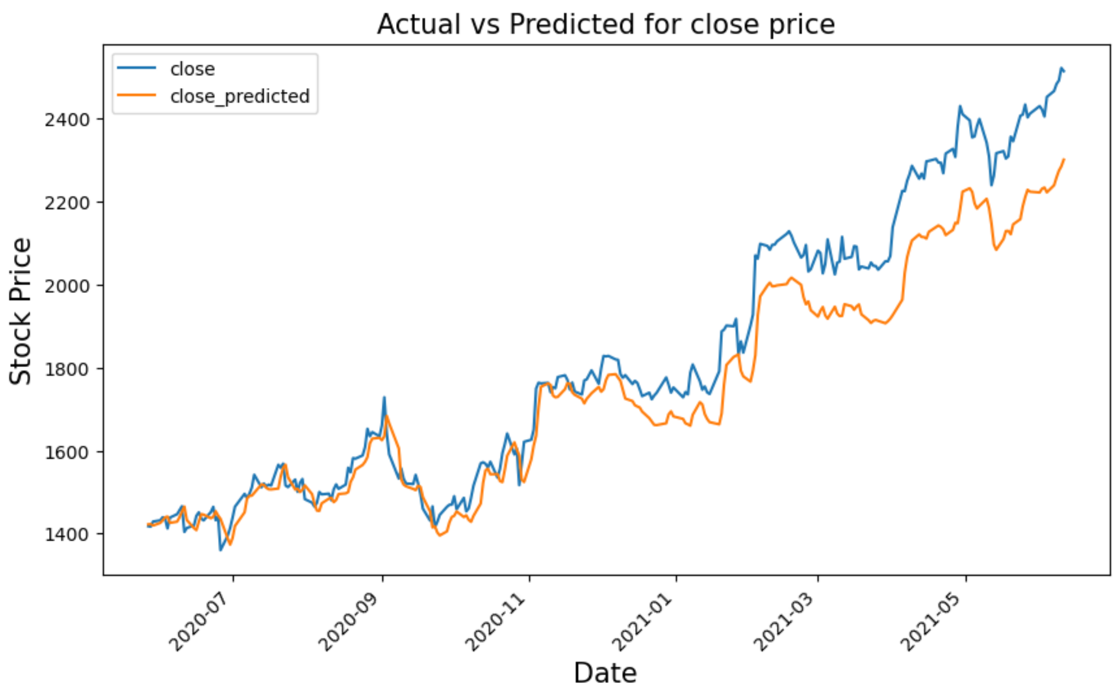
[22]:

```
# plotting the actual open and predicted open prices on date index
df_merge[['open','open_predicted']].plot(figsize=(10,6))
plt.xticks(rotation=45)
plt.xlabel('Date',size=15)
plt.ylabel('Stock Price',size=15)
plt.title('Actual vs Predicted for open price',size=15)
plt.show()
```



[23]:

```
# plotting the actual close and predicted close prices on date index
df_merge[['close','close_predicted']].plot(figsize=(10,6))
plt.xticks(rotation=45)
plt.xlabel('Date',size=15)
plt.ylabel('Stock Price',size=15)
plt.title('Actual vs Predicted for close price',size=15)
plt.show()
```



[24]:
```
# Creating a dataframe and adding 10 days to existing index

df_merge = df_merge.append(pd.DataFrame(columns=df_merge.columns,
                                        index=pd.date_range(start=df_merge.
 ↪index[-1], periods=11, freq='D', closed='right')))
df_merge['2021-06-09':'2021-06-16']
```

[24]:

	open	close	open_predicted	close_predicted
2021-06-09	2499.50	2491.40	2332.436279	2274.111816
2021-06-10	2494.01	2521.60	2344.484863	2284.591309
2021-06-11	2524.92	2513.93	2361.345459	2300.645508
2021-06-12	NaN	NaN	NaN	NaN
2021-06-13	NaN	NaN	NaN	NaN
2021-06-14	NaN	NaN	NaN	NaN
2021-06-15	NaN	NaN	NaN	NaN
2021-06-16	NaN	NaN	NaN	NaN

[25]:
```
# creating a DataFrame and filling values of open and close column
upcoming_prediction = pd.DataFrame(columns=['open','close'],index=df_merge.
 ↪index)
upcoming_prediction.index=pd.to_datetime(upcoming_prediction.index)
```

[26]:
```
curr_seq = test_seq[-1:]
```

```python
for i in range(-10,0):
  up_pred = model.predict(curr_seq)
  upcoming_prediction.iloc[i] = up_pred
  curr_seq = np.append(curr_seq[0][1:],up_pred,axis=0)
  curr_seq = curr_seq.reshape(test_seq[-1:].shape)
```

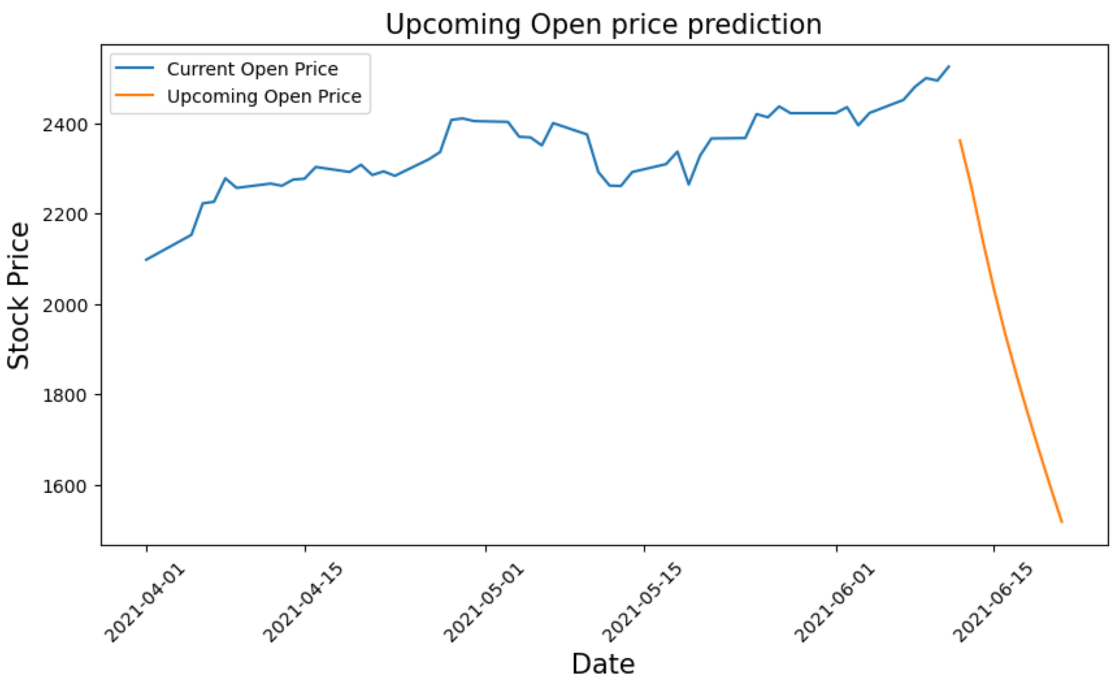
```
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 59ms/step
```

[27]:

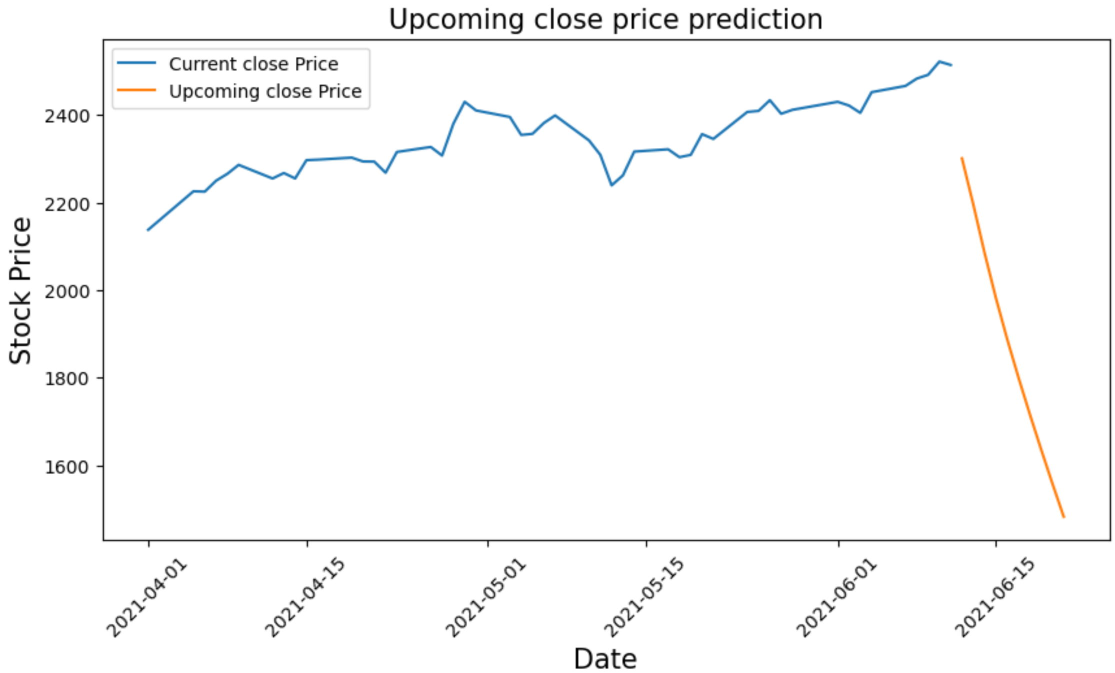
```python
# inversing Normalization/scaling
upcoming_prediction[['open','close']] = MMS.
 ↪inverse_transform(upcoming_prediction[['open','close']])
```

[28]:

```python
# plotting Upcoming Open price on date index
fig,ax=plt.subplots(figsize=(10,5))
ax.plot(df_merge.loc['2021-04-01':,'open'],label='Current Open Price')
ax.plot(upcoming_prediction.loc['2021-04-01':,'open'],label='Upcoming Open␣
 ↪Price')
plt.setp(ax.xaxis.get_majorticklabels(), rotation=45)
ax.set_xlabel('Date',size=15)
ax.set_ylabel('Stock Price',size=15)
ax.set_title('Upcoming Open price prediction',size=15)
ax.legend()
fig.show()
```



[29]:
```
# plotting Upcoming Close price on date index
fig,ax=plt.subplots(figsize=(10,5))
ax.plot(df_merge.loc['2021-04-01':,'close'],label='Current close Price')
ax.plot(upcoming_prediction.loc['2021-04-01':,'close'],label='Upcoming close␣
↪Price')
plt.setp(ax.xaxis.get_majorticklabels(), rotation=45)
ax.set_xlabel('Date',size=15)
ax.set_ylabel('Stock Price',size=15)
ax.set_title('Upcoming close price prediction',size=15)
ax.legend()
fig.show()
```



Upcoming close price prediction
Current close Price
Upcoming close Price
Stock Price
2400
2200
2000
1800
1600
Date
2021-04-01
2021-04-15
2021-05-01
2021-05-15
2021-06-01
2021-06-15