lstm

September 23, 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
```

[22]:
```
df = pd.read_csv(r'C:\Users\HP\Downloads\GOOG.csv') # data_importing
df.head(10) # fetching first 10 rows of dataset
```

[22]:
```
  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
```

[23]:
```python
# shape of data
print("Shape of data:",df.shape)
```

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

[24]:
```python
# statistical description of data
df.describe()
```

[24]:
```
             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
```

[25]:
```python
# 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
```

[26]:
```python
# checking null values
df.isnull().sum()
```

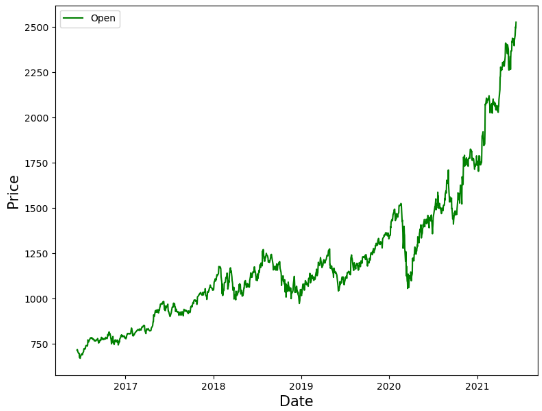
[26]:
```
symbol         0
date           0
close          0
high           0
low            0
open           0
volume         0
adjClose       0
adjHigh        0
adjLow         0
adjOpen        0
adjVolume      0
divCash        0
```

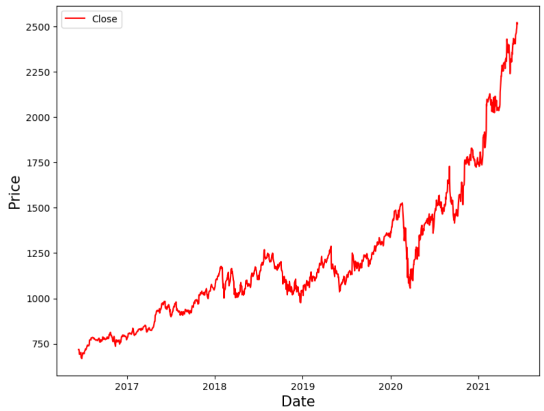
```
splitFactor    0
dtype: int64
```

[29]:
```python
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)
```

[29]:
```
              open   close
date
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
```

[32]:
```python
# 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()
```





[33]:
```python
# normalizing all the values of all columns using MinMaxScaler
MMS = MinMaxScaler()
df[df.columns] = MMS.fit_transform(df)
df.head(10)
```

[33]:
```
                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
```

[34]:
```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
```

[34]: 944

[35]:
```python
train_data = df[:training_size]
test_data  = df[training_size:]

train_data.shape, test_data.shape
```

[35]: ((944, 2), (314, 2))

[36]:
```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))
```

[37]:
```
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
```

[37]: ((894, 50, 2), (894, 2), (264, 50, 2), (264, 2))

[38]:
```
# 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_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 50, 50)            10600

 dropout_1 (Dropout)         (None, 50, 50)            0

 lstm_3 (LSTM)               (None, 50)                20200

 dense_1 (Dense)             (None, 2)                 102

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

[39]:
```
# 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 [==============================] - 7s 79ms/step - loss: 0.0108 -
mean_absolute_error: 0.0738 - val_loss: 0.0302 - val_mean_absolute_error: 0.1484
```

```
Epoch 2/100
28/28 [==============================] - 1s 44ms/step - loss: 0.0011 -
mean_absolute_error: 0.0264 - val_loss: 0.0059 - val_mean_absolute_error: 0.0595
Epoch 3/100
28/28 [==============================] - 1s 43ms/step - loss: 5.3247e-04 -
mean_absolute_error: 0.0170 - val_loss: 0.0062 - val_mean_absolute_error: 0.0640
Epoch 4/100
28/28 [==============================] - 1s 48ms/step - loss: 4.8688e-04 -
mean_absolute_error: 0.0162 - val_loss: 0.0074 - val_mean_absolute_error: 0.0700
Epoch 5/100
28/28 [==============================] - 1s 45ms/step - loss: 4.7569e-04 -
mean_absolute_error: 0.0162 - val_loss: 0.0058 - val_mean_absolute_error: 0.0605
Epoch 6/100
28/28 [==============================] - 1s 40ms/step - loss: 4.7943e-04 -
mean_absolute_error: 0.0159 - val_loss: 0.0047 - val_mean_absolute_error: 0.0531
Epoch 7/100
28/28 [==============================] - 1s 45ms/step - loss: 4.4376e-04 -
mean_absolute_error: 0.0155 - val_loss: 0.0049 - val_mean_absolute_error: 0.0547
Epoch 8/100
28/28 [==============================] - 1s 53ms/step - loss: 4.2734e-04 -
mean_absolute_error: 0.0151 - val_loss: 0.0059 - val_mean_absolute_error: 0.0609
Epoch 9/100
28/28 [==============================] - 1s 37ms/step - loss: 4.2596e-04 -
mean_absolute_error: 0.0153 - val_loss: 0.0045 - val_mean_absolute_error: 0.0519
Epoch 10/100
28/28 [==============================] - 1s 43ms/step - loss: 4.0932e-04 -
mean_absolute_error: 0.0146 - val_loss: 0.0042 - val_mean_absolute_error: 0.0500
Epoch 11/100
28/28 [==============================] - 1s 48ms/step - loss: 4.4156e-04 -
mean_absolute_error: 0.0155 - val_loss: 0.0035 - val_mean_absolute_error: 0.0453
Epoch 12/100
28/28 [==============================] - 1s 40ms/step - loss: 3.9894e-04 -
mean_absolute_error: 0.0146 - val_loss: 0.0044 - val_mean_absolute_error: 0.0509
Epoch 13/100
28/28 [==============================] - 1s 47ms/step - loss: 3.8052e-04 -
mean_absolute_error: 0.0142 - val_loss: 0.0061 - val_mean_absolute_error: 0.0626
Epoch 14/100
28/28 [==============================] - 2s 58ms/step - loss: 3.9301e-04 -
mean_absolute_error: 0.0145 - val_loss: 0.0053 - val_mean_absolute_error: 0.0571
Epoch 15/100
28/28 [==============================] - 1s 51ms/step - loss: 3.6701e-04 -
mean_absolute_error: 0.0141 - val_loss: 0.0045 - val_mean_absolute_error: 0.0531
Epoch 16/100
28/28 [==============================] - 2s 78ms/step - loss: 3.7793e-04 -
mean_absolute_error: 0.0140 - val_loss: 0.0062 - val_mean_absolute_error: 0.0626
Epoch 17/100
28/28 [==============================] - 1s 45ms/step - loss: 3.6938e-04 -
mean_absolute_error: 0.0139 - val_loss: 0.0073 - val_mean_absolute_error: 0.0703
```

```
Epoch 18/100
28/28 [==============================] - 1s 38ms/step - loss: 3.6832e-04 -
mean_absolute_error: 0.0140 - val_loss: 0.0045 - val_mean_absolute_error: 0.0527
Epoch 19/100
28/28 [==============================] - 1s 40ms/step - loss: 3.6783e-04 -
mean_absolute_error: 0.0139 - val_loss: 0.0038 - val_mean_absolute_error: 0.0480
Epoch 20/100
28/28 [==============================] - 1s 47ms/step - loss: 3.5116e-04 -
mean_absolute_error: 0.0136 - val_loss: 0.0073 - val_mean_absolute_error: 0.0699
Epoch 21/100
28/28 [==============================] - 1s 42ms/step - loss: 3.3116e-04 -
mean_absolute_error: 0.0134 - val_loss: 0.0049 - val_mean_absolute_error: 0.0559
Epoch 22/100
28/28 [==============================] - 1s 42ms/step - loss: 3.3223e-04 -
mean_absolute_error: 0.0133 - val_loss: 0.0046 - val_mean_absolute_error: 0.0533
Epoch 23/100
28/28 [==============================] - 2s 58ms/step - loss: 3.0748e-04 -
mean_absolute_error: 0.0128 - val_loss: 0.0038 - val_mean_absolute_error: 0.0476
Epoch 24/100
28/28 [==============================] - 2s 54ms/step - loss: 3.5453e-04 -
mean_absolute_error: 0.0139 - val_loss: 0.0065 - val_mean_absolute_error: 0.0663
Epoch 25/100
28/28 [==============================] - 1s 38ms/step - loss: 3.3464e-04 -
mean_absolute_error: 0.0132 - val_loss: 0.0038 - val_mean_absolute_error: 0.0485
Epoch 26/100
28/28 [==============================] - 1s 38ms/step - loss: 3.0547e-04 -
mean_absolute_error: 0.0128 - val_loss: 0.0042 - val_mean_absolute_error: 0.0514
Epoch 27/100
28/28 [==============================] - 1s 41ms/step - loss: 3.0356e-04 -
mean_absolute_error: 0.0127 - val_loss: 0.0071 - val_mean_absolute_error: 0.0695
Epoch 28/100
28/28 [==============================] - 1s 42ms/step - loss: 2.8202e-04 -
mean_absolute_error: 0.0124 - val_loss: 0.0046 - val_mean_absolute_error: 0.0527
Epoch 29/100
28/28 [==============================] - 1s 46ms/step - loss: 3.0122e-04 -
mean_absolute_error: 0.0129 - val_loss: 0.0077 - val_mean_absolute_error: 0.0734
Epoch 30/100
28/28 [==============================] - 1s 38ms/step - loss: 2.9285e-04 -
mean_absolute_error: 0.0126 - val_loss: 0.0033 - val_mean_absolute_error: 0.0434
Epoch 31/100
28/28 [==============================] - 1s 44ms/step - loss: 2.8927e-04 -
mean_absolute_error: 0.0124 - val_loss: 0.0074 - val_mean_absolute_error: 0.0717
Epoch 32/100
28/28 [==============================] - 1s 44ms/step - loss: 2.8645e-04 -
mean_absolute_error: 0.0125 - val_loss: 0.0029 - val_mean_absolute_error: 0.0413
Epoch 33/100
28/28 [==============================] - 1s 44ms/step - loss: 2.6631e-04 -
mean_absolute_error: 0.0121 - val_loss: 0.0058 - val_mean_absolute_error: 0.0610
```

```
Epoch 34/100
28/28 [==============================] - 1s 43ms/step - loss: 2.6602e-04 -
mean_absolute_error: 0.0120 - val_loss: 0.0049 - val_mean_absolute_error: 0.0551
Epoch 35/100
28/28 [==============================] - 1s 44ms/step - loss: 2.5153e-04 -
mean_absolute_error: 0.0117 - val_loss: 0.0044 - val_mean_absolute_error: 0.0507
Epoch 36/100
28/28 [==============================] - 1s 41ms/step - loss: 2.6420e-04 -
mean_absolute_error: 0.0119 - val_loss: 0.0036 - val_mean_absolute_error: 0.0456
Epoch 37/100
28/28 [==============================] - 1s 41ms/step - loss: 2.6043e-04 -
mean_absolute_error: 0.0118 - val_loss: 0.0040 - val_mean_absolute_error: 0.0491
Epoch 38/100
28/28 [==============================] - 1s 41ms/step - loss: 2.4631e-04 -
mean_absolute_error: 0.0115 - val_loss: 0.0052 - val_mean_absolute_error: 0.0570
Epoch 39/100
28/28 [==============================] - 1s 37ms/step - loss: 2.4459e-04 -
mean_absolute_error: 0.0114 - val_loss: 0.0042 - val_mean_absolute_error: 0.0500
Epoch 40/100
28/28 [==============================] - 1s 46ms/step - loss: 2.4589e-04 -
mean_absolute_error: 0.0112 - val_loss: 0.0045 - val_mean_absolute_error: 0.0518
Epoch 41/100
28/28 [==============================] - 1s 39ms/step - loss: 2.2968e-04 -
mean_absolute_error: 0.0111 - val_loss: 0.0057 - val_mean_absolute_error: 0.0599
Epoch 42/100
28/28 [==============================] - 1s 46ms/step - loss: 2.2330e-04 -
mean_absolute_error: 0.0109 - val_loss: 0.0048 - val_mean_absolute_error: 0.0543
Epoch 43/100
28/28 [==============================] - 1s 46ms/step - loss: 2.2874e-04 -
mean_absolute_error: 0.0110 - val_loss: 0.0052 - val_mean_absolute_error: 0.0570
Epoch 44/100
28/28 [==============================] - 1s 44ms/step - loss: 2.3879e-04 -
mean_absolute_error: 0.0114 - val_loss: 0.0045 - val_mean_absolute_error: 0.0511
Epoch 45/100
28/28 [==============================] - 1s 40ms/step - loss: 2.3402e-04 -
mean_absolute_error: 0.0111 - val_loss: 0.0055 - val_mean_absolute_error: 0.0596
Epoch 46/100
28/28 [==============================] - 1s 42ms/step - loss: 2.2058e-04 -
mean_absolute_error: 0.0108 - val_loss: 0.0034 - val_mean_absolute_error: 0.0435
Epoch 47/100
28/28 [==============================] - 1s 39ms/step - loss: 2.1834e-04 -
mean_absolute_error: 0.0105 - val_loss: 0.0023 - val_mean_absolute_error: 0.0358
Epoch 48/100
28/28 [==============================] - 1s 44ms/step - loss: 2.5983e-04 -
mean_absolute_error: 0.0120 - val_loss: 0.0020 - val_mean_absolute_error: 0.0327
Epoch 49/100
28/28 [==============================] - 1s 43ms/step - loss: 2.1299e-04 -
mean_absolute_error: 0.0107 - val_loss: 0.0039 - val_mean_absolute_error: 0.0490
```

```
Epoch 50/100
28/28 [==============================] - 1s 41ms/step - loss: 1.9429e-04 -
mean_absolute_error: 0.0101 - val_loss: 0.0038 - val_mean_absolute_error: 0.0464
Epoch 51/100
28/28 [==============================] - 1s 41ms/step - loss: 2.0890e-04 -
mean_absolute_error: 0.0106 - val_loss: 0.0044 - val_mean_absolute_error: 0.0513
Epoch 52/100
28/28 [==============================] - 1s 38ms/step - loss: 2.0151e-04 -
mean_absolute_error: 0.0103 - val_loss: 0.0040 - val_mean_absolute_error: 0.0491
Epoch 53/100
28/28 [==============================] - 1s 46ms/step - loss: 2.2244e-04 -
mean_absolute_error: 0.0110 - val_loss: 0.0041 - val_mean_absolute_error: 0.0498
Epoch 54/100
28/28 [==============================] - 1s 40ms/step - loss: 2.1931e-04 -
mean_absolute_error: 0.0110 - val_loss: 0.0023 - val_mean_absolute_error: 0.0349
Epoch 55/100
28/28 [==============================] - 1s 40ms/step - loss: 1.8771e-04 -
mean_absolute_error: 0.0099 - val_loss: 0.0038 - val_mean_absolute_error: 0.0493
Epoch 56/100
28/28 [==============================] - 1s 38ms/step - loss: 1.7298e-04 -
mean_absolute_error: 0.0097 - val_loss: 0.0024 - val_mean_absolute_error: 0.0365
Epoch 57/100
28/28 [==============================] - 1s 38ms/step - loss: 1.9629e-04 -
mean_absolute_error: 0.0101 - val_loss: 0.0019 - val_mean_absolute_error: 0.0322
Epoch 58/100
28/28 [==============================] - 1s 40ms/step - loss: 1.7890e-04 -
mean_absolute_error: 0.0097 - val_loss: 0.0032 - val_mean_absolute_error: 0.0439
Epoch 59/100
28/28 [==============================] - 1s 43ms/step - loss: 1.8177e-04 -
mean_absolute_error: 0.0098 - val_loss: 0.0016 - val_mean_absolute_error: 0.0294
Epoch 60/100
28/28 [==============================] - 1s 49ms/step - loss: 1.7654e-04 -
mean_absolute_error: 0.0096 - val_loss: 0.0028 - val_mean_absolute_error: 0.0407
Epoch 61/100
28/28 [==============================] - 1s 41ms/step - loss: 1.9551e-04 -
mean_absolute_error: 0.0101 - val_loss: 0.0034 - val_mean_absolute_error: 0.0465
Epoch 62/100
28/28 [==============================] - 1s 41ms/step - loss: 1.6233e-04 -
mean_absolute_error: 0.0092 - val_loss: 0.0018 - val_mean_absolute_error: 0.0319
Epoch 63/100
28/28 [==============================] - 1s 43ms/step - loss: 1.6966e-04 -
mean_absolute_error: 0.0094 - val_loss: 0.0033 - val_mean_absolute_error: 0.0444
Epoch 64/100
28/28 [==============================] - 1s 38ms/step - loss: 1.8138e-04 -
mean_absolute_error: 0.0098 - val_loss: 0.0038 - val_mean_absolute_error: 0.0490
Epoch 65/100
28/28 [==============================] - 1s 45ms/step - loss: 1.5972e-04 -
mean_absolute_error: 0.0093 - val_loss: 0.0024 - val_mean_absolute_error: 0.0366
```

```
Epoch 66/100
28/28 [==============================] - 1s 40ms/step - loss: 1.6331e-04 -
mean_absolute_error: 0.0093 - val_loss: 0.0021 - val_mean_absolute_error: 0.0348
Epoch 67/100
28/28 [==============================] - 1s 38ms/step - loss: 1.6296e-04 -
mean_absolute_error: 0.0093 - val_loss: 0.0024 - val_mean_absolute_error: 0.0371
Epoch 68/100
28/28 [==============================] - 1s 37ms/step - loss: 1.6735e-04 -
mean_absolute_error: 0.0094 - val_loss: 0.0020 - val_mean_absolute_error: 0.0343
Epoch 69/100
28/28 [==============================] - 1s 42ms/step - loss: 1.5119e-04 -
mean_absolute_error: 0.0090 - val_loss: 0.0036 - val_mean_absolute_error: 0.0493
Epoch 70/100
28/28 [==============================] - 1s 43ms/step - loss: 1.6336e-04 -
mean_absolute_error: 0.0092 - val_loss: 9.9312e-04 - val_mean_absolute_error:
0.0233
Epoch 71/100
28/28 [==============================] - 1s 43ms/step - loss: 1.6590e-04 -
mean_absolute_error: 0.0094 - val_loss: 8.5798e-04 - val_mean_absolute_error:
0.0216
Epoch 72/100
28/28 [==============================] - 1s 42ms/step - loss: 1.4698e-04 -
mean_absolute_error: 0.0088 - val_loss: 0.0010 - val_mean_absolute_error: 0.0241
Epoch 73/100
28/28 [==============================] - 1s 42ms/step - loss: 1.4955e-04 -
mean_absolute_error: 0.0088 - val_loss: 0.0017 - val_mean_absolute_error: 0.0323
Epoch 74/100
28/28 [==============================] - 1s 42ms/step - loss: 1.5718e-04 -
mean_absolute_error: 0.0091 - val_loss: 0.0032 - val_mean_absolute_error: 0.0458
Epoch 75/100
28/28 [==============================] - 1s 39ms/step - loss: 1.4633e-04 -
mean_absolute_error: 0.0087 - val_loss: 0.0020 - val_mean_absolute_error: 0.0340
Epoch 76/100
28/28 [==============================] - 1s 38ms/step - loss: 1.6043e-04 -
mean_absolute_error: 0.0092 - val_loss: 9.8429e-04 - val_mean_absolute_error:
0.0231
Epoch 77/100
28/28 [==============================] - 1s 38ms/step - loss: 1.4806e-04 -
mean_absolute_error: 0.0089 - val_loss: 0.0030 - val_mean_absolute_error: 0.0455
Epoch 78/100
28/28 [==============================] - 1s 38ms/step - loss: 1.5536e-04 -
mean_absolute_error: 0.0089 - val_loss: 0.0026 - val_mean_absolute_error: 0.0391
Epoch 79/100
28/28 [==============================] - 1s 40ms/step - loss: 1.3401e-04 -
mean_absolute_error: 0.0083 - val_loss: 0.0018 - val_mean_absolute_error: 0.0332
Epoch 80/100
28/28 [==============================] - 1s 41ms/step - loss: 1.3263e-04 -
mean_absolute_error: 0.0083 - val_loss: 8.1906e-04 - val_mean_absolute_error:
```

```
0.0213
Epoch 81/100
28/28 [==============================] - 1s 42ms/step - loss: 1.2922e-04 -
mean_absolute_error: 0.0081 - val_loss: 0.0010 - val_mean_absolute_error: 0.0242
Epoch 82/100
28/28 [==============================] - 1s 48ms/step - loss: 1.3665e-04 -
mean_absolute_error: 0.0084 - val_loss: 0.0015 - val_mean_absolute_error: 0.0287
Epoch 83/100
28/28 [==============================] - 1s 42ms/step - loss: 1.4627e-04 -
mean_absolute_error: 0.0088 - val_loss: 5.4069e-04 - val_mean_absolute_error:
0.0178
Epoch 84/100
28/28 [==============================] - 1s 40ms/step - loss: 1.6317e-04 -
mean_absolute_error: 0.0095 - val_loss: 5.5351e-04 - val_mean_absolute_error:
0.0176
Epoch 85/100
28/28 [==============================] - 1s 39ms/step - loss: 1.3886e-04 -
mean_absolute_error: 0.0087 - val_loss: 0.0014 - val_mean_absolute_error: 0.0283
Epoch 86/100
28/28 [==============================] - 1s 42ms/step - loss: 1.2551e-04 -
mean_absolute_error: 0.0082 - val_loss: 0.0024 - val_mean_absolute_error: 0.0404
Epoch 87/100
28/28 [==============================] - 1s 42ms/step - loss: 1.3378e-04 -
mean_absolute_error: 0.0083 - val_loss: 9.0971e-04 - val_mean_absolute_error:
0.0224
Epoch 88/100
28/28 [==============================] - 1s 38ms/step - loss: 1.1468e-04 -
mean_absolute_error: 0.0077 - val_loss: 0.0013 - val_mean_absolute_error: 0.0264
Epoch 89/100
28/28 [==============================] - 1s 39ms/step - loss: 1.2304e-04 -
mean_absolute_error: 0.0079 - val_loss: 0.0015 - val_mean_absolute_error: 0.0290
Epoch 90/100
28/28 [==============================] - 1s 38ms/step - loss: 1.3755e-04 -
mean_absolute_error: 0.0085 - val_loss: 5.5682e-04 - val_mean_absolute_error:
0.0179
Epoch 91/100
28/28 [==============================] - 1s 39ms/step - loss: 1.3929e-04 -
mean_absolute_error: 0.0083 - val_loss: 0.0018 - val_mean_absolute_error: 0.0331
Epoch 92/100
28/28 [==============================] - 1s 40ms/step - loss: 1.2723e-04 -
mean_absolute_error: 0.0080 - val_loss: 0.0028 - val_mean_absolute_error: 0.0408
Epoch 93/100
28/28 [==============================] - 1s 43ms/step - loss: 1.3525e-04 -
mean_absolute_error: 0.0084 - val_loss: 0.0019 - val_mean_absolute_error: 0.0323
Epoch 94/100
28/28 [==============================] - 1s 41ms/step - loss: 1.2579e-04 -
mean_absolute_error: 0.0082 - val_loss: 0.0021 - val_mean_absolute_error: 0.0350
Epoch 95/100
```

```
28/28 [==============================] - 1s 37ms/step - loss: 1.2792e-04 -
mean_absolute_error: 0.0084 - val_loss: 0.0016 - val_mean_absolute_error: 0.0295
Epoch 96/100
28/28 [==============================] - 1s 40ms/step - loss: 1.2170e-04 -
mean_absolute_error: 0.0078 - val_loss: 0.0012 - val_mean_absolute_error: 0.0253
Epoch 97/100
28/28 [==============================] - 1s 41ms/step - loss: 1.2382e-04 -
mean_absolute_error: 0.0079 - val_loss: 7.9888e-04 - val_mean_absolute_error:
0.0197
Epoch 98/100
28/28 [==============================] - 1s 36ms/step - loss: 1.1421e-04 -
mean_absolute_error: 0.0078 - val_loss: 0.0015 - val_mean_absolute_error: 0.0294
Epoch 99/100
28/28 [==============================] - 1s 38ms/step - loss: 1.1097e-04 -
mean_absolute_error: 0.0074 - val_loss: 0.0020 - val_mean_absolute_error: 0.0354
Epoch 100/100
28/28 [==============================] - 1s 47ms/step - loss: 1.1377e-04 -
mean_absolute_error: 0.0076 - val_loss: 4.7172e-04 - val_mean_absolute_error:
0.0161
```

[39]:
```
<keras.src.callbacks.History at 0x1a20016ac40>
```

[40]:
```python
# predicting the values after running the model
test_predicted = model.predict(test_seq)
test_predicted[:5]
```

```
9/9 [==============================] - 1s 14ms/step
```

[40]:
```
array([[0.41237482, 0.41365182],
       [0.4132493 , 0.41397762],
       [0.41134813, 0.4116734 ],
       [0.41388828, 0.41503865],
       [0.41699478, 0.41866353]], dtype=float32)
```

[41]:
```python
# Inversing normalization/scaling on predicted data
test_inverse_predicted = MMS.inverse_transform(test_predicted)
test_inverse_predicted[:5]
```

[41]:
```
array([[1435.5099, 1434.8975],
       [1437.1312, 1435.5013],
       [1433.6064, 1431.2307],
       [1438.3158, 1437.4678],
       [1444.075 , 1444.1859]], dtype=float32)
```

[42]:
```python
# 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'],
```

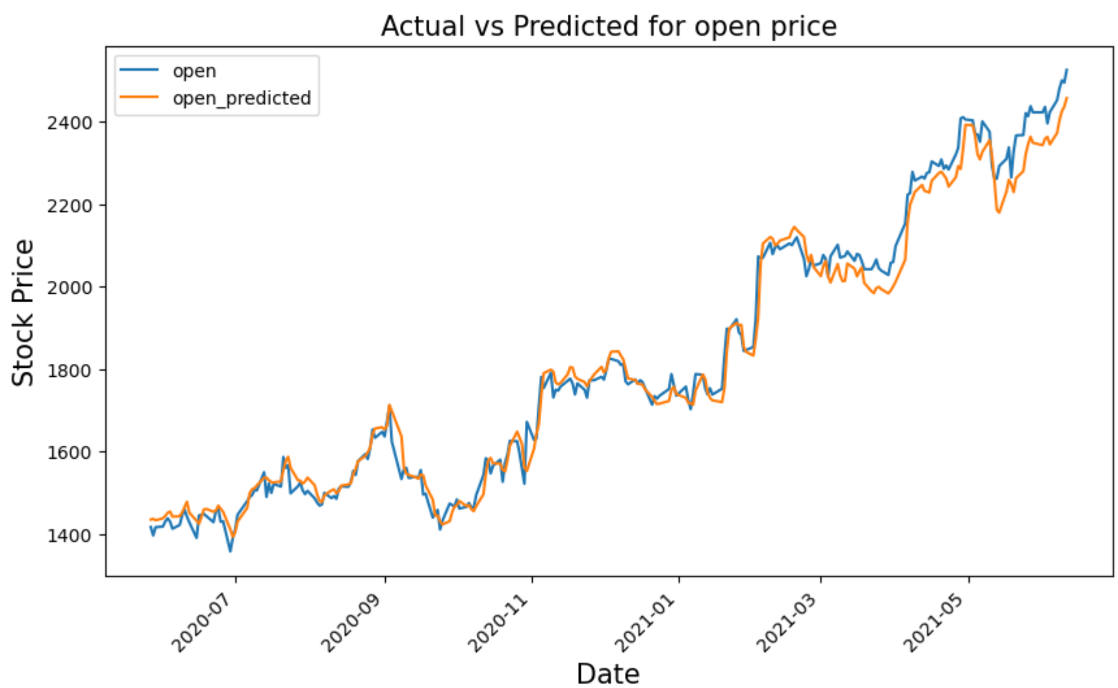
```
                                          index=df.iloc[-264:].index)], axis=1)
```

[43]:
```
# Inversing normalization/scaling
df_merge[['open','close']] = MMS.inverse_transform(df_merge[['open','close']])
df_merge.head()
```

[43]:

date	open	close	open_predicted	close_predicted
2020-05-27	1417.25	1417.84	1435.509888	1434.897461
2020-05-28	1396.86	1416.73	1437.131226	1435.501343
2020-05-29	1416.94	1428.92	1433.606445	1431.230713
2020-06-01	1418.39	1431.82	1438.315796	1437.467773
2020-06-02	1430.55	1439.22	1444.074951	1444.185913

[44]:
```
# 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()
```



[45]:
```
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']
```

[45]:

	open	close	open_predicted	close_predicted
2021-06-09	2499.50	2491.40	2424.801514	2556.667236
2021-06-10	2494.01	2521.60	2435.927002	2566.250977
2021-06-11	2524.92	2513.93	2456.750244	2588.945312
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

[46]:
```
# 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)
```

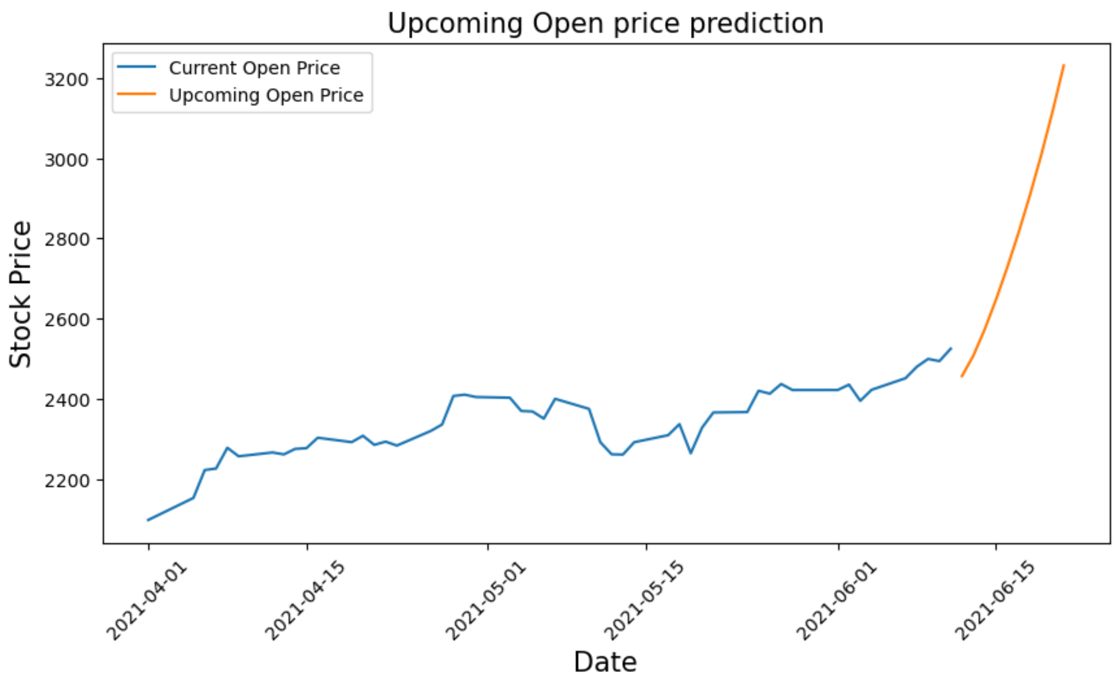
[47]:
```
curr_seq = test_seq[-1:]

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 37ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 33ms/step
```

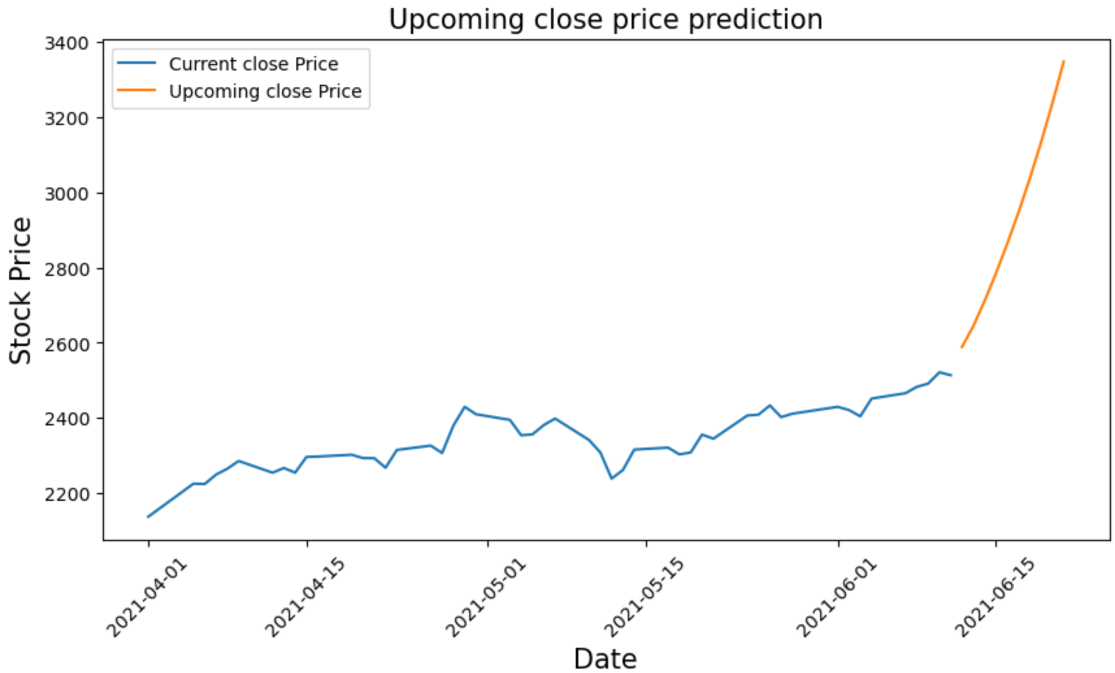
[48]:
```
# inversing Normalization/scaling
upcoming_prediction[['open','close']] = MMS.
 ↪inverse_transform(upcoming_prediction[['open','close']])
```

[49]:
```
# 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()
```



[50]:
```
# 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
3400
3200
3000
2800
2600
2400
2200
2021-04-01
2021-04-15
2021-05-01
2021-05-15
2021-06-01
2021-06-15
Date

[ ]: