In [3]:
```python
#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
```

In [5]:
```python
df = pd.read_csv('GOOG.csv') # data_importing
df.head(10) # fetching first 10 rows of dataset
```

Out[5]:

	symbol	date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFactor
0	GOOG	2016-06-14 00:00:00+00:00	718.27	722.47	713.1200	716.48	1306065	718.27	722.47	713.1200	716.48	1306065	0.0	1.0
1	GOOG	2016-06-15 00:00:00+00:00	718.92	722.98	717.3100	719.00	1214517	718.92	722.98	717.3100	719.00	1214517	0.0	1.0
2	GOOG	2016-06-16 00:00:00+00:00	710.36	716.65	703.2600	714.91	1982471	710.36	716.65	703.2600	714.91	1982471	0.0	1.0
3	GOOG	2016-06-17 00:00:00+00:00	691.72	708.82	688.4515	708.65	3402357	691.72	708.82	688.4515	708.65	3402357	0.0	1.0
4	GOOG	2016-06-20 00:00:00+00:00	693.71	702.48	693.4100	698.77	2082538	693.71	702.48	693.4100	698.77	2082538	0.0	1.0
5	GOOG	2016-06-21 00:00:00+00:00	695.94	702.77	692.0100	698.40	1465634	695.94	702.77	692.0100	698.40	1465634	0.0	1.0
6	GOOG	2016-06-22 00:00:00+00:00	697.46	700.86	693.0819	699.06	1184318	697.46	700.86	693.0819	699.06	1184318	0.0	1.0
7	GOOG	2016-06-23 00:00:00+00:00	701.87	701.95	687.0000	697.45	2171415	701.87	701.95	687.0000	697.45	2171415	0.0	1.0
8	GOOG	2016-06-24 00:00:00+00:00	675.22	689.40	673.4500	675.17	4449022	675.22	689.40	673.4500	675.17	4449022	0.0	1.0
9	GOOG	2016-06-27 00:00:00+00:00	668.26	672.30	663.2840	671.00	2641085	668.26	672.30	663.2840	671.00	2641085	0.0	1.0

In [6]:
```python
# shape of data
print("Shape of data:",df.shape)
```

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

In [7]:
```python
# statistical description of data
df.describe()
```

Out[7]:

	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash	splitFactor
count	1258.000000	1258.000000	1258.000000	1258.000000	1.258000e+03	1258.000000	1258.000000	1258.000000	1258.000000	1.258000e+03	1258.0	1258.0
mean	1216.317067	1227.430934	1204.176430	1215.260779	1.601590e+06	1216.317067	1227.430936	1204.176436	1215.260779	1.601590e+06	0.0	1.0
std	383.333358	387.570872	378.777094	382.446995	6.960172e+05	383.333358	387.570873	378.777099	382.446995	6.960172e+05	0.0	0.0
min	668.260000	672.300000	663.284000	671.000000	3.467530e+05	668.260000	672.300000	663.284000	671.000000	3.467530e+05	0.0	1.0
25%	960.802500	968.757500	952.182500	959.005000	1.173522e+06	960.802500	968.757500	952.182500	959.005000	1.173522e+06	0.0	1.0
50%	1132.460000	1143.935000	1117.915000	1131.150000	1.412588e+06	1132.460000	1143.935000	1117.915000	1131.150000	1.412588e+06	0.0	1.0
75%	1360.595000	1374.345000	1348.557500	1361.075000	1.812156e+06	1360.595000	1374.345000	1348.557500	1361.075000	1.812156e+06	0.0	1.0
max	2521.600000	2526.990000	2498.290000	2524.920000	6.207027e+06	2521.600000	2526.990000	2498.290000	2524.920000	6.207027e+06	0.0	1.0

In [8]:
```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
```

In [9]:
```python
# checking null values
df.isnull().sum()
```

Out[9]:
```
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
```

In [10]:
```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)
```

Out[10]:

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

In [11]:
```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()
```

In [12]:
```python
# normalizing all the values of all columns using MinMaxScaler
MMS = MinMaxScaler()
df[df.columns] = MMS.fit_transform(df)
df.head(10)
```

Out[12]:

date	open	close
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

In [13]:
```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
```

Out[13]:
```
944
```

In [14]:
```python
train_data = df[:training_size]
test_data  = df[training_size:]

train_data.shape, test_data.shape
```

Out[14]:
```
((944, 2), (314, 2))
```

In [15]:
```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))
```

In [16]:
```python
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
```

Out[16]:
```
((894, 50, 2), (894, 2), (264, 50, 2), (264, 2))
```

In [17]:
```python
# 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)
_________________________________________________________________
```

In [18]:
```python
# 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 [==============================] - 22s 166ms/step - loss: 0.0097 - mean_absolute_error: 0.0685 - val_loss: 0.0255 - val_mean_absolute_error: 0.1371
Epoch 2/100
28/28 [==============================] - 2s 82ms/step - loss: 8.8686e-04 - mean_absolute_error: 0.0239 - val_loss: 0.0054 - val_mean_absolute_error: 0.0564
Epoch 3/100
28/28 [==============================] - 2s 83ms/step - loss: 4.7381e-04 - mean_absolute_error: 0.0162 - val_loss: 0.0041 - val_mean_absolute_error: 0.0488
Epoch 4/100
28/28 [==============================] - 2s 83ms/step - loss: 4.5095e-04 - mean_absolute_error: 0.0155 - val_loss: 0.0055 - val_mean_absolute_error: 0.0568
Epoch 5/100
28/28 [==============================] - 2s 80ms/step - loss: 4.4740e-04 - mean_absolute_error: 0.0154 - val_loss: 0.0045 - val_mean_absolute_error: 0.0505
Epoch 6/100
28/28 [==============================] - 2s 79ms/step - loss: 4.3762e-04 - mean_absolute_error: 0.0153 - val_loss: 0.0036 - val_mean_absolute_error: 0.0450
Epoch 7/100
28/28 [==============================] - 2s 81ms/step - loss: 4.0935e-04 - mean_absolute_error: 0.0148 - val_loss: 0.0067 - val_mean_absolute_error: 0.0660
Epoch 8/100
28/28 [==============================] - 2s 80ms/step - loss: 4.5618e-04 - mean_absolute_error: 0.0155 - val_loss: 0.0047 - val_mean_absolute_error: 0.0527
Epoch 9/100
28/28 [==============================] - 2s 82ms/step - loss: 4.0863e-04 - mean_absolute_error: 0.0147 - val_loss: 0.0046 - val_mean_absolute_error: 0.0526
Epoch 10/100
28/28 [==============================] - 2s 83ms/step - loss: 3.9101e-04 - mean_absolute_error: 0.0144 - val_loss: 0.0044 - val_mean_absolute_error: 0.0506
Epoch 11/100
28/28 [==============================] - 2s 83ms/step - loss: 3.7201e-04 - mean_absolute_error: 0.0141 - val_loss: 0.0040 - val_mean_absolute_error: 0.0484
Epoch 12/100
28/28 [==============================] - 2s 80ms/step - loss: 4.0585e-04 - mean_absolute_error: 0.0146 - val_loss: 0.0056 - val_mean_absolute_error: 0.0603
Epoch 13/100
28/28 [==============================] - 2s 83ms/step - loss: 4.0111e-04 - mean_absolute_error: 0.0145 - val_loss: 0.0050 - val_mean_absolute_error: 0.0560
Epoch 14/100
28/28 [==============================] - 2s 82ms/step - loss: 3.7329e-04 - mean_absolute_error: 0.0141 - val_loss: 0.0071 - val_mean_absolute_error: 0.0694
Epoch 15/100
28/28 [==============================] - 2s 82ms/step - loss: 3.6214e-04 - mean_absolute_error: 0.0139 - val_loss: 0.0046 - val_mean_absolute_error: 0.0540
Epoch 16/100
28/28 [==============================] - 2s 82ms/step - loss: 3.7578e-04 - mean_absolute_error: 0.0140 - val_loss: 0.0056 - val_mean_absolute_error: 0.0608
Epoch 17/100
28/28 [==============================] - 2s 83ms/step - loss: 3.5818e-04 - mean_absolute_error: 0.0137 - val_loss: 0.0037 - val_mean_absolute_error: 0.0460
Epoch 18/100
28/28 [==============================] - 2s 86ms/step - loss: 3.4563e-04 - mean_absolute_error: 0.0137 - val_loss: 0.0029 - val_mean_absolute_error: 0.0409
Epoch 19/100
28/28 [==============================] - 2s 82ms/step - loss: 3.3414e-04 - mean_absolute_error: 0.0134 - val_loss: 0.0051 - val_mean_absolute_error: 0.0564
Epoch 20/100
28/28 [==============================] - 2s 80ms/step - loss: 3.1890e-04 - mean_absolute_error: 0.0132 - val_loss: 0.0035 - val_mean_absolute_error: 0.0451
Epoch 21/100
28/28 [==============================] - 2s 81ms/step - loss: 3.0953e-04 - mean_absolute_error: 0.0130 - val_loss: 0.0047 - val_mean_absolute_error: 0.0552
Epoch 22/100
28/28 [==============================] - 2s 83ms/step - loss: 3.1646e-04 - mean_absolute_error: 0.0130 - val_loss: 0.0046 - val_mean_absolute_error: 0.0525
Epoch 23/100
28/28 [==============================] - 2s 81ms/step - loss: 2.9800e-04 - mean_absolute_error: 0.0126 - val_loss: 0.0036 - val_mean_absolute_error: 0.0456
Epoch 24/100
28/28 [==============================] - 2s 81ms/step - loss: 3.1065e-04 - mean_absolute_error: 0.0130 - val_loss: 0.0043 - val_mean_absolute_error: 0.0512
Epoch 25/100
28/28 [==============================] - 2s 82ms/step - loss: 2.9578e-04 - mean_absolute_error: 0.0124 - val_loss: 0.0067 - val_mean_absolute_error: 0.0668
Epoch 26/100
28/28 [==============================] - 2s 84ms/step - loss: 2.8085e-04 - mean_absolute_error: 0.0123 - val_loss: 0.0040 - val_mean_absolute_error: 0.0489
Epoch 27/100
28/28 [==============================] - 2s 80ms/step - loss: 2.8881e-04 - mean_absolute_error: 0.0125 - val_loss: 0.0057 - val_mean_absolute_error: 0.0603
Epoch 28/100
28/28 [==============================] - 2s 79ms/step - loss: 2.8040e-04 - mean_absolute_error: 0.0124 - val_loss: 0.0046 - val_mean_absolute_error: 0.0537
Epoch 29/100
28/28 [==============================] - 2s 82ms/step - loss: 2.9441e-04 - mean_absolute_error: 0.0127 - val_loss: 0.0038 - val_mean_absolute_error: 0.0473
Epoch 30/100
28/28 [==============================] - 2s 82ms/step - loss: 2.8119e-04 - mean_absolute_error: 0.0125 - val_loss: 0.0044 - val_mean_absolute_error: 0.0518
Epoch 31/100
28/28 [==============================] - 2s 81ms/step - loss: 2.9603e-04 - mean_absolute_error: 0.0127 - val_loss: 0.0048 - val_mean_absolute_error: 0.0551
Epoch 32/100
28/28 [==============================] - 2s 81ms/step - loss: 2.7141e-04 - mean_absolute_error: 0.0123 - val_loss: 0.0055 - val_mean_absolute_error: 0.0602
Epoch 33/100
28/28 [==============================] - 2s 80ms/step - loss: 2.8152e-04 - mean_absolute_error: 0.0124 - val_loss: 0.0043 - val_mean_absolute_error: 0.0519
Epoch 34/100
28/28 [==============================] - 2s 82ms/step - loss: 2.5571e-04 - mean_absolute_error: 0.0116 - val_loss: 0.0037 - val_mean_absolute_error: 0.0482
Epoch 35/100
28/28 [==============================] - 2s 80ms/step - loss: 2.5481e-04 - mean_absolute_error: 0.0120 - val_loss: 0.0052 - val_mean_absolute_error: 0.0588
Epoch 36/100
28/28 [==============================] - 2s 82ms/step - loss: 2.5568e-04 - mean_absolute_error: 0.0120 - val_loss: 0.0051 - val_mean_absolute_error: 0.0584
Epoch 37/100
28/28 [==============================] - 2s 79ms/step - loss: 2.4937e-04 - mean_absolute_error: 0.0117 - val_loss: 0.0039 - val_mean_absolute_error: 0.0494
Epoch 38/100
28/28 [==============================] - 2s 78ms/step - loss: 2.3182e-04 - mean_absolute_error: 0.0112 - val_loss: 0.0047 - val_mean_absolute_error: 0.0558
Epoch 39/100
28/28 [==============================] - 2s 79ms/step - loss: 2.4406e-04 - mean_absolute_error: 0.0114 - val_loss: 0.0063 - val_mean_absolute_error: 0.0645
Epoch 40/100
28/28 [==============================] - 2s 81ms/step - loss: 2.4822e-04 - mean_absolute_error: 0.0116 - val_loss: 0.0030 - val_mean_absolute_error: 0.0410
Epoch 41/100
28/28 [==============================] - 2s 87ms/step - loss: 2.3073e-04 - mean_absolute_error: 0.0110 - val_loss: 0.0040 - val_mean_absolute_error: 0.0501
Epoch 42/100
28/28 [==============================] - 2s 82ms/step - loss: 2.3640e-04 - mean_absolute_error: 0.0113 - val_loss: 0.0021 - val_mean_absolute_error: 0.0340
Epoch 43/100
28/28 [==============================] - 2s 85ms/step - loss: 2.3616e-04 - mean_absolute_error: 0.0112 - val_loss: 0.0047 - val_mean_absolute_error: 0.0540
Epoch 44/100
28/28 [==============================] - 2s 80ms/step - loss: 2.4587e-04 - mean_absolute_error: 0.0113 - val_loss: 0.0030 - val_mean_absolute_error: 0.0420
Epoch 45/100
28/28 [==============================] - 2s 78ms/step - loss: 2.3272e-04 - mean_absolute_error: 0.0112 - val_loss: 0.0028 - val_mean_absolute_error: 0.0399
Epoch 46/100
28/28 [==============================] - 2s 81ms/step - loss: 2.3185e-04 - mean_absolute_error: 0.0112 - val_loss: 0.0052 - val_mean_absolute_error: 0.0588
Epoch 47/100
28/28 [==============================] - 2s 82ms/step - loss: 2.2773e-04 - mean_absolute_error: 0.0109 - val_loss: 0.0040 - val_mean_absolute_error: 0.0504
Epoch 48/100
28/28 [==============================] - 2s 83ms/step - loss: 2.1620e-04 - mean_absolute_error: 0.0106 - val_loss: 0.0036 - val_mean_absolute_error: 0.0476
Epoch 49/100
28/28 [==============================] - 2s 82ms/step - loss: 2.0136e-04 - mean_absolute_error: 0.0103 - val_loss: 0.0024 - val_mean_absolute_error: 0.0373
Epoch 50/100
28/28 [==============================] - 2s 85ms/step - loss: 2.0978e-04 - mean_absolute_error: 0.0104 - val_loss: 0.0048 - val_mean_absolute_error: 0.0561
Epoch 51/100
28/28 [==============================] - 2s 82ms/step - loss: 2.1407e-04 - mean_absolute_error: 0.0106 - val_loss: 0.0026 - val_mean_absolute_error: 0.0396
Epoch 52/100
28/28 [==============================] - 2s 82ms/step - loss: 1.9279e-04 - mean_absolute_error: 0.0102 - val_loss: 0.0030 - val_mean_absolute_error: 0.0424
Epoch 53/100
28/28 [==============================] - 2s 83ms/step - loss: 2.0450e-04 - mean_absolute_error: 0.0105 - val_loss: 0.0028 - val_mean_absolute_error: 0.0413
Epoch 54/100
28/28 [==============================] - 2s 82ms/step - loss: 1.8474e-04 - mean_absolute_error: 0.0098 - val_loss: 0.0037 - val_mean_absolute_error: 0.0486
Epoch 55/100
28/28 [==============================] - 2s 80ms/step - loss: 1.8209e-04 - mean_absolute_error: 0.0097 - val_loss: 0.0030 - val_mean_absolute_error: 0.0439
Epoch 56/100
28/28 [==============================] - 2s 83ms/step - loss: 1.8713e-04 - mean_absolute_error: 0.0099 - val_loss: 0.0015 - val_mean_absolute_error: 0.0286
Epoch 57/100
28/28 [==============================] - 2s 83ms/step - loss: 2.2317e-04 - mean_absolute_error: 0.0111 - val_loss: 0.0029 - val_mean_absolute_error: 0.0420
Epoch 58/100
28/28 [==============================] - 2s 83ms/step - loss: 1.8750e-04 - mean_absolute_error: 0.0101 - val_loss: 0.0039 - val_mean_absolute_error: 0.0517
Epoch 59/100
28/28 [==============================] - 2s 83ms/step - loss: 1.9172e-04 - mean_absolute_error: 0.0102 - val_loss: 0.0024 - val_mean_absolute_error: 0.0392
Epoch 60/100
28/28 [==============================] - 2s 83ms/step - loss: 1.7319e-04 - mean_absolute_error: 0.0095 - val_loss: 0.0019 - val_mean_absolute_error: 0.0344
Epoch 61/100
28/28 [==============================] - 2s 79ms/step - loss: 1.7522e-04 - mean_absolute_error: 0.0095 - val_loss: 0.0016 - val_mean_absolute_error: 0.0304
Epoch 62/100
28/28 [==============================] - 2s 84ms/step - loss: 1.6226e-04 - mean_absolute_error: 0.0092 - val_loss: 0.0013 - val_mean_absolute_error: 0.0270
Epoch 63/100
28/28 [==============================] - 2s 83ms/step - loss: 1.8802e-04 - mean_absolute_error: 0.0101 - val_loss: 0.0018 - val_mean_absolute_error: 0.0325
Epoch 64/100
28/28 [==============================] - 2s 82ms/step - loss: 1.8689e-04 - mean_absolute_error: 0.0101 - val_loss: 0.0031 - val_mean_absolute_error: 0.0455
Epoch 65/100
28/28 [==============================] - 2s 82ms/step - loss: 1.5050e-04 - mean_absolute_error: 0.0089 - val_loss: 0.0032 - val_mean_absolute_error: 0.0472
Epoch 66/100
28/28 [==============================] - 2s 83ms/step - loss: 1.4651e-04 - mean_absolute_error: 0.0087 - val_loss: 0.0025 - val_mean_absolute_error: 0.0413
Epoch 67/100
28/28 [==============================] - 2s 83ms/step - loss: 1.7537e-04 - mean_absolute_error: 0.0095 - val_loss: 0.0031 - val_mean_absolute_error: 0.0460
Epoch 68/100
28/28 [==============================] - 2s 82ms/step - loss: 1.6437e-04 - mean_absolute_error: 0.0092 - val_loss: 0.0030 - val_mean_absolute_error: 0.0447
Epoch 69/100
28/28 [==============================] - 2s 83ms/step - loss: 1.4881e-04 - mean_absolute_error: 0.0087 - val_loss: 0.0032 - val_mean_absolute_error: 0.0475
Epoch 70/100
28/28 [==============================] - 2s 82ms/step - loss: 1.4630e-04 - mean_absolute_error: 0.0087 - val_loss: 0.0013 - val_mean_absolute_error: 0.0273
Epoch 71/100
28/28 [==============================] - 2s 80ms/step - loss: 1.4351e-04 - mean_absolute_error: 0.0088 - val_loss: 0.0028 - val_mean_absolute_error: 0.0439
Epoch 72/100
28/28 [==============================] - 2s 83ms/step - loss: 1.3683e-04 - mean_absolute_error: 0.0084 - val_loss: 0.0024 - val_mean_absolute_error: 0.0399
Epoch 73/100
28/28 [==============================] - 2s 84ms/step - loss: 1.4092e-04 - mean_absolute_error: 0.0085 - val_loss: 0.0039 - val_mean_absolute_error: 0.0525
Epoch 74/100
28/28 [==============================] - 2s 80ms/step - loss: 1.4551e-04 - mean_absolute_error: 0.0088 - val_loss: 0.0036 - val_mean_absolute_error: 0.0506
Epoch 75/100
28/28 [==============================] - 2s 81ms/step - loss: 1.3759e-04 - mean_absolute_error: 0.0085 - val_loss: 0.0030 - val_mean_absolute_error: 0.0457
Epoch 76/100
28/28 [==============================] - 2s 82ms/step - loss: 1.2677e-04 - mean_absolute_error: 0.0081 - val_loss: 0.0029 - val_mean_absolute_error: 0.0436
Epoch 77/100
28/28 [==============================] - 2s 80ms/step - loss: 1.1410e-04 - mean_absolute_error: 0.0077 - val_loss: 0.0035 - val_mean_absolute_error: 0.0506
Epoch 78/100
28/28 [==============================] - 2s 80ms/step - loss: 1.4284e-04 - mean_absolute_error: 0.0087 - val_loss: 0.0027 - val_mean_absolute_error: 0.0423
Epoch 79/100
28/28 [==============================] - 2s 81ms/step - loss: 1.2746e-04 - mean_absolute_error: 0.0081 - val_loss: 0.0020 - val_mean_absolute_error: 0.0368
Epoch 80/100
28/28 [==============================] - 2s 82ms/step - loss: 1.2077e-04 - mean_absolute_error: 0.0079 - val_loss: 0.0025 - val_mean_absolute_error: 0.0407
Epoch 81/100
28/28 [==============================] - 2s 79ms/step - loss: 1.2284e-04 - mean_absolute_error: 0.0080 - val_loss: 0.0015 - val_mean_absolute_error: 0.0302
Epoch 82/100
28/28 [==============================] - 2s 82ms/step - loss: 1.1973e-04 - mean_absolute_error: 0.0079 - val_loss: 0.0014 - val_mean_absolute_error: 0.0300
Epoch 83/100
28/28 [==============================] - 2s 82ms/step - loss: 1.1957e-04 - mean_absolute_error: 0.0078 - val_loss: 0.0014 - val_mean_absolute_error: 0.0293
Epoch 84/100
28/28 [==============================] - 2s 83ms/step - loss: 1.1097e-04 - mean_absolute_error: 0.0075 - val_loss: 0.0026 - val_mean_absolute_error: 0.0420
Epoch 85/100
28/28 [==============================] - 2s 82ms/step - loss: 1.1278e-04 - mean_absolute_error: 0.0076 - val_loss: 0.0025 - val_mean_absolute_error: 0.0400
Epoch 86/100
28/28 [==============================] - 2s 80ms/step - loss: 1.3644e-04 - mean_absolute_error: 0.0084 - val_loss: 0.0043 - val_mean_absolute_error: 0.0564
Epoch 87/100
28/28 [==============================] - 2s 82ms/step - loss: 1.3914e-04 - mean_absolute_error: 0.0087 - val_loss: 0.0022 - val_mean_absolute_error: 0.0378
Epoch 88/100
28/28 [==============================] - 2s 81ms/step - loss: 1.2593e-04 - mean_absolute_error: 0.0080 - val_loss: 0.0014 - val_mean_absolute_error: 0.0295
Epoch 89/100
28/28 [==============================] - 2s 81ms/step - loss: 1.2309e-04 - mean_absolute_error: 0.0079 - val_loss: 0.0017 - val_mean_absolute_error: 0.0328
Epoch 90/100
28/28 [==============================] - 2s 80ms/step - loss: 1.1345e-04 - mean_absolute_error: 0.0076 - val_loss: 0.0026 - val_mean_absolute_error: 0.0431
Epoch 91/100
28/28 [==============================] - 2s 82ms/step - loss: 1.3374e-04 - mean_absolute_error: 0.0086 - val_loss: 0.0024 - val_mean_absolute_error: 0.0415
Epoch 92/100
28/28 [==============================] - 2s 83ms/step - loss: 1.1313e-04 - mean_absolute_error: 0.0076 - val_loss: 0.0015 - val_mean_absolute_error: 0.0303
Epoch 93/100
28/28 [==============================] - 2s 83ms/step - loss: 1.1582e-04 - mean_absolute_error: 0.0077 - val_loss: 0.0018 - val_mean_absolute_error: 0.0347
Epoch 94/100
28/28 [==============================] - 2s 77ms/step - loss: 1.0109e-04 - mean_absolute_error: 0.0072 - val_loss: 0.0015 - val_mean_absolute_error: 0.0307
Epoch 95/100
28/28 [==============================] - 2s 79ms/step - loss: 1.0930e-04 - mean_absolute_error: 0.0075 - val_loss: 0.0018 - val_mean_absolute_error: 0.0339
Epoch 96/100
28/28 [==============================] - 2s 80ms/step - loss: 9.5683e-05 - mean_absolute_error: 0.0069 - val_loss: 0.0011 - val_mean_absolute_error: 0.0264
Epoch 97/100
28/28 [==============================] - 2s 86ms/step - loss: 9.8859e-05 - mean_absolute_error: 0.0069 - val_loss: 0.0013 - val_mean_absolute_error: 0.0288
Epoch 98/100
28/28 [==============================] - 2s 84ms/step - loss: 1.0392e-04 - mean_absolute_error: 0.0072 - val_loss: 0.0013 - val_mean_absolute_error: 0.0276
Epoch 99/100
28/28 [==============================] - 2s 85ms/step - loss: 9.7757e-05 - mean_absolute_error: 0.0070 - val_loss: 0.0010 - val_mean_absolute_error: 0.0244
Epoch 100/100
28/28 [==============================] - 2s 82ms/step - loss: 1.0613e-04 - mean_absolute_error: 0.0073 - val_loss: 0.0023 - val_mean_absolute_error: 0.0402
```

Out[18]:
```
<keras.src.callbacks.History at 0x19af1b2c850>
```

In [19]:
```python
# predicting the values after running the model
test_predicted = model.predict(test_seq)
test_predicted[:5]
```

```
9/9 [==============================] - 4s 27ms/step
```

Out[19]:
```
array([[0.39780027, 0.40200517],
       [0.39793885, 0.40237296],
       [0.39566642, 0.40026984],
       [0.3998409 , 0.40395623],
       [0.40322605, 0.40731996]], dtype=float32)
```

In [20]:
```python
# Inversing normalization/scaling on predicted data
test_inverse_predicted = MMS.inverse_transform(test_predicted)
test_inverse_predicted[:5]
```

Out[20]:
```
array([[1408.4899, 1413.3123],
       [1408.7468, 1413.9939],
       [1404.5339, 1410.0961],
       [1412.2731, 1416.9283],
       [1418.5488, 1423.1625]], dtype=float32)
```

In [21]:
```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)
```

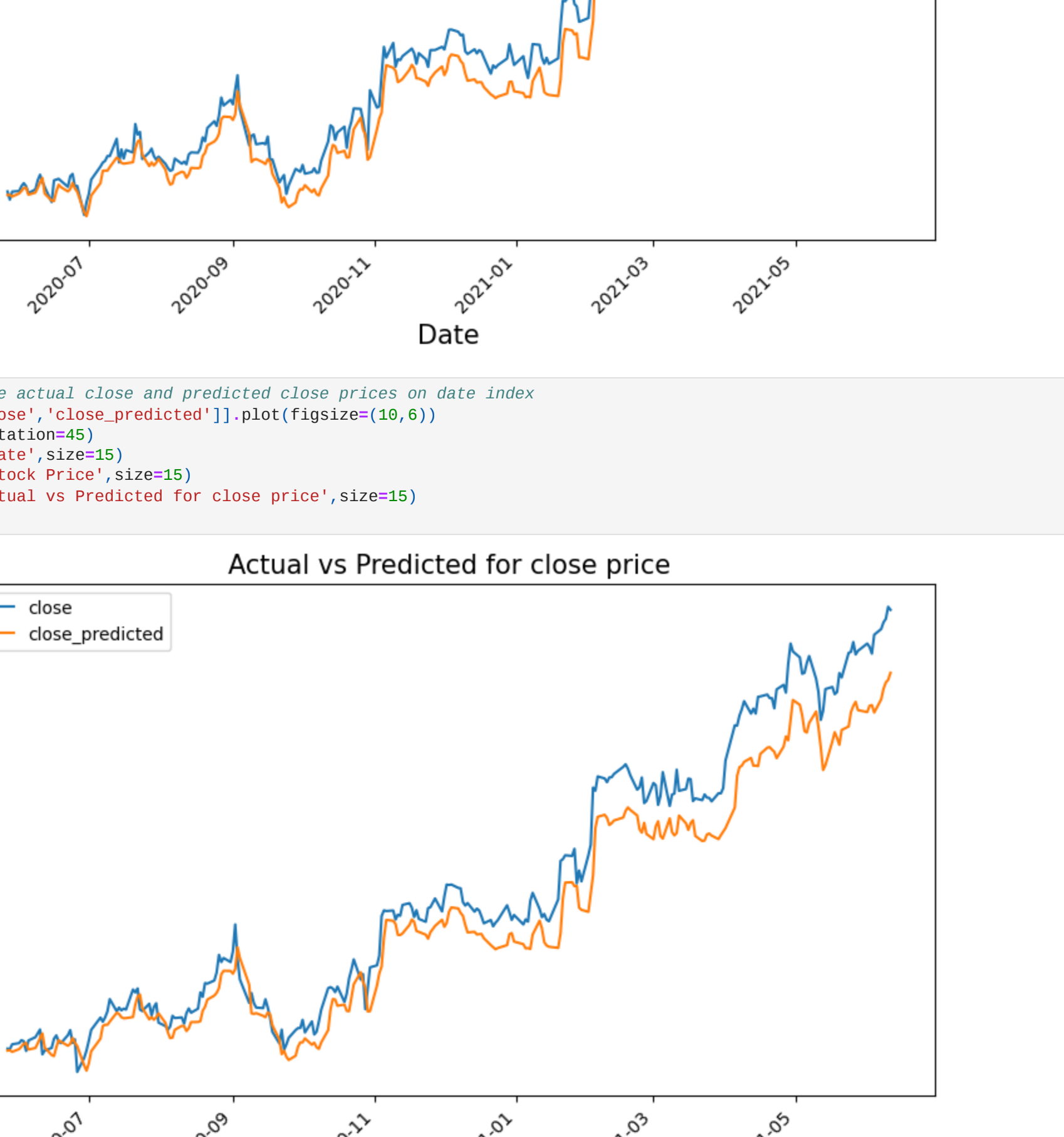
In [22]:
```python
# Inversing normalization/scaling
df_merge[['open','close']] = MMS.inverse_transform(df_merge[['open','close']])
df_merge.head()
```

Out[22]:

date	open	close	open_predicted	close_predicted
2020-05-27	1417.25	1417.84	1408.489868	1413.312256
2020-05-28	1396.86	1416.73	1408.746826	1413.993896
2020-05-29	1416.94	1428.92	1404.533936	1410.096069
2020-06-01	1418.39	1431.82	1412.273071	1416.928345
2020-06-02	1430.55	1439.22	1418.548828	1423.162476

In [23]:
```python
# 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()
```

In [24]:
```python
# 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()
```



In [25]:
```python
# 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']
```

Out[25]:

	open	close	open_predicted	close_predicted
2021-06-09	2499.50	2491.40	2327.616455	2332.594971
2021-06-10	2494.01	2521.60	2332.955078	2339.145508
2021-06-11	2524.92	2513.93	2352.908203	2356.723389
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

In [26]:
```python
# 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)
```

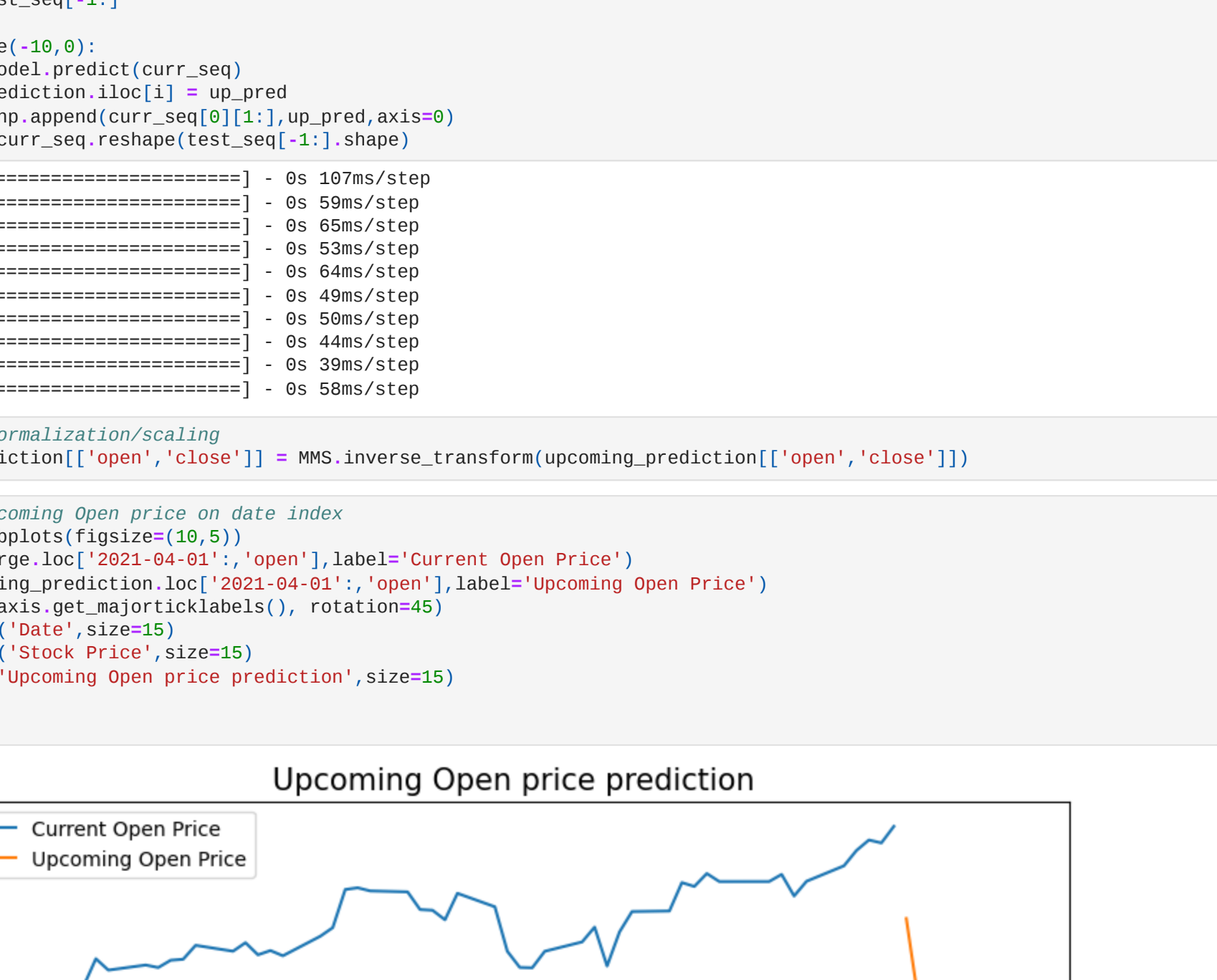
In [27]:
```python
curr_seq = test_seq[-1:]

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)
```

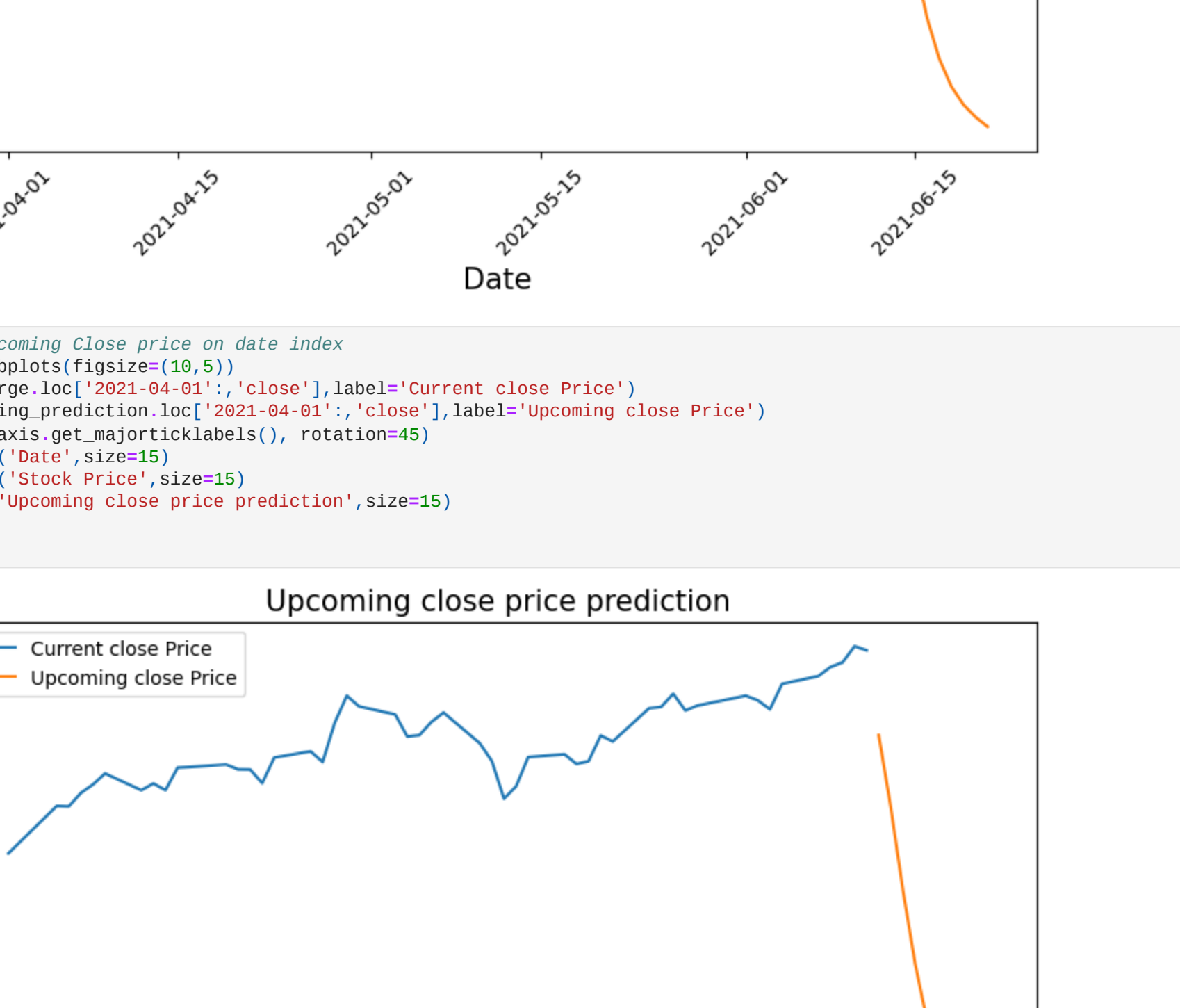
```
1/1 [==============================] - 0s 107ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 58ms/step
```

In [28]:
```python
# inversing Normalization/scaling
upcoming_prediction[['open','close']] = MMS.inverse_transform(upcoming_prediction[['open','close']])
```

In [29]:
```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()
```



In [30]:
```python
# 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()
```



In [ ]: