Google-stock-price-prediction

October 4, 2023

1 GOOGLE STOCK PRICE PREDICTION USING LSTM MODEL

[202]:

```python
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import os
import math
import datetime as dt

from sklearn.preprocessing import MinMaxScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.layers import LSTM
```

[141]:

```python
#Loading Dataset
df=pd.read_csv('GOOG.csv')
```

[142]:

```python
#loading head & tail
df.head()
```

[142]:

```
         Date        Open        High         Low       Close   Adj Close  \
0  2022-10-03   97.220001   99.970001   97.019997   99.300003   99.300003
1  2022-10-04  101.040001  102.720001  101.040001  102.410004  102.410004
2  2022-10-05  100.690002  102.739998   99.739998  102.220001  102.220001
3  2022-10-06  101.500000  103.730003  101.500000  102.239998  102.239998
4  2022-10-07  100.650002  101.419998   99.209999   99.570000   99.570000

     Volume
0  24840000
1  22580900
2  18475500
3  17156200
```

```
4  24249900
```

```
[143]: df.tail()
```

```
[143]:          Date        Open        High         Low       Close   Adj Close  \
       246  2023-09-26  130.914001  131.404999  128.190002  129.449997  129.449997
       247  2023-09-27  129.440002  131.720001  129.380005  131.460007  131.460007
       248  2023-09-28  130.690002  134.179993  130.690002  133.130005  133.130005
       249  2023-09-29  134.080002  134.889999  131.320007  131.850006  131.850006
       250  2023-10-02  132.154999  135.360001  132.065002  135.169998  135.169998

              Volume
       246  20378800
       247  18764200
       248  18201400
       249  23224200
       250  19189000
```

```
[149]: data_training = df[df['Date']<'2019-01-01'].copy()
       data_training
```

```
[149]: Empty DataFrame
       Columns: [Date, Open, High, Low, Close, Adj Close, Volume]
       Index: []
```

```
[150]: data_test = df[df['Date']>='2019-01-01'].copy()
       data_test
```

```
[150]:          Date        Open        High         Low       Close   Adj Close  \
       0    2022-10-03   97.220001   99.970001   97.019997   99.300003   99.300003
       1    2022-10-04  101.040001  102.720001  101.040001  102.410004  102.410004
       2    2022-10-05  100.690002  102.739998   99.739998  102.220001  102.220001
       3    2022-10-06  101.500000  103.730003  101.500000  102.239998  102.239998
       4    2022-10-07  100.650002  101.419998   99.209999   99.570000   99.570000
       ..          …           …           …           …           …           …
       246  2023-09-26  130.914001  131.404999  128.190002  129.449997  129.449997
       247  2023-09-27  129.440002  131.720001  129.380005  131.460007  131.460007
       248  2023-09-28  130.690002  134.179993  130.690002  133.130005  133.130005
       249  2023-09-29  134.080002  134.889999  131.320007  131.850006  131.850006
       250  2023-10-02  132.154999  135.360001  132.065002  135.169998  135.169998

              Volume
       0    24840000
       1    22580900
       2    18475500
       3    17156200
       4    24249900
```

```
..         …
246  20378800
247  18764200
248  18201400
249  23224200
250  19189000

[251 rows x 7 columns]
```

[151]:
```
df.shape
```

[151]:
```
(251, 7)
```

[152]:
```
#Removing duplicate values
df = df.loc[~df.index.duplicated(keep='first')]
```

[153]:
```
print('Total no. of days present in dataset:',df.shape[0])
print('Total no. of fields present in dataset:',df.shape[1])
```

```
Total no. of days present in dataset: 251
Total no. of fields present in dataset: 7
```

[154]:
```
df.shape
```

[154]:
```
(251, 7)
```

[155]:
```
df.head()
```

[155]:
```
         Date        Open        High         Low       Close   Adj Close  \
0  2022-10-03   97.220001   99.970001   97.019997   99.300003   99.300003
1  2022-10-04  101.040001  102.720001  101.040001  102.410004  102.410004
2  2022-10-05  100.690002  102.739998   99.739998  102.220001  102.220001
3  2022-10-06  101.500000  103.730003  101.500000  102.239998  102.239998
4  2022-10-07  100.650002  101.419998   99.209999   99.570000   99.570000

     Volume
0  24840000
1  22580900
2  18475500
3  17156200
4  24249900
```

[156]:
```
df
```

[156]:
```
           Date        Open        High         Low       Close   Adj Close  \
0    2022-10-03   97.220001   99.970001   97.019997   99.300003   99.300003
1    2022-10-04  101.040001  102.720001  101.040001  102.410004  102.410004
2    2022-10-05  100.690002  102.739998   99.739998  102.220001  102.220001
```

```
3    2022-10-06  101.500000  103.730003  101.500000  102.239998  102.239998
4    2022-10-07  100.650002  101.419998   99.209999   99.570000   99.570000
..          …           …           …           …           …           …
246  2023-09-26  130.914001  131.404999  128.190002  129.449997  129.449997
247  2023-09-27  129.440002  131.720001  129.380005  131.460007  131.460007
248  2023-09-28  130.690002  134.179993  130.690002  133.130005  133.130005
249  2023-09-29  134.080002  134.889999  131.320007  131.850006  131.850006
250  2023-10-02  132.154999  135.360001  132.065002  135.169998  135.169998

       Volume
0    24840000
1    22580900
2    18475500
3    17156200
4    24249900
..        …
246  20378800
247  18764200
248  18201400
249  23224200
250  19189000

[251 rows x 7 columns]
```

[157]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 251 entries, 0 to 250
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       251 non-null    object
 1   Open       251 non-null    float64
 2   High       251 non-null    float64
 3   Low        251 non-null    float64
 4   Close      251 non-null    float64
 5   Adj Close  251 non-null    float64
 6   Volume     251 non-null    int64
dtypes: float64(5), int64(1), object(1)
memory usage: 15.7+ KB
```

[158]:
```
df.describe(include="all")
```

[158]:
```
              Date        Open        High         Low       Close  \
count          251  251.000000  251.000000  251.000000  251.000000
unique         251         NaN         NaN         NaN         NaN
top     2022-10-03         NaN         NaN         NaN         NaN
freq             1         NaN         NaN         NaN         NaN
```

```
mean           NaN  109.362311  110.875104  108.213757  109.571574
std            NaN   15.791147   15.829224   15.795351   15.806917
min            NaN   85.510002   86.550003   83.449997   83.489998
25%            NaN   95.759998   97.349998   94.470001   95.840000
50%            NaN  105.230003  106.540001  104.209999  105.120003
75%            NaN  123.972499  125.444999  122.895001  124.215000
max            NaN  138.830002  139.929993  137.630005  138.990005

         Adj Close        Volume
count   251.000000  2.510000e+02
unique         NaN           NaN
top            NaN           NaN
freq           NaN           NaN
mean    109.571574  2.666822e+07
std      15.806917  1.108759e+07
min      83.489998  8.567800e+06
25%      95.840000  2.009725e+07
50%     105.120003  2.365610e+07
75%     124.215000  3.004110e+07
max     138.990005  9.779860e+07
```

[159]:

```
#checking null values
print('Null Values:',df.isnull().values.sum())
```

```
Null Values: 0
```

[161]:

```
training_data = df.drop(['Date', 'Adj Close'], axis = 1)
training_data.head()
```

[161]:

```
         Open        High         Low       Close    Volume
0   97.220001   99.970001   97.019997   99.300003  24840000
1  101.040001  102.720001  101.040001  102.410004  22580900
2  100.690002  102.739998   99.739998  102.220001  18475500
3  101.500000  103.730003  101.500000  102.239998  17156200
4  100.650002  101.419998   99.209999   99.570000  24249900
```

[162]:

```
scaler = MinMaxScaler()
training_data = scaler.fit_transform(training_data)
training_data
```

[162]:

```
array([[0.21961739, 0.25140503, 0.25046139, 0.28486492, 0.1823608 ],
       [0.2912603 , 0.30292246, 0.32465857, 0.34090097, 0.15704331],
       [0.28469617, 0.30329708, 0.30066443, 0.33747749, 0.11103453],
       …,
       [0.84733683, 0.89228173, 0.87190842, 0.89441443, 0.10796272],
       [0.91091523, 0.90558271, 0.88353641, 0.87135139, 0.1642527 ],
       [0.8748124 , 0.91438754, 0.89728678, 0.93117105, 0.11903065]])
```

[163]:

```
x_train = []
y_train = []
```

[164]:

```
training_data.shape[0]
```

[164]:

```
251
```

[165]:

```
for i in range(60, training_data.shape[0]):
    x_train.append(training_data[i-60:i])
    y_train.append(training_data[i, 0])
```

[166]:

```
x_train , y_train = np.array(x_train), np.array(y_train)
```

[167]:

```
#rows,columns,dimensions
x_train.shape
```

[167]:

```
(191, 60, 5)
```

[168]:

```
y_train.shape
```

[168]:

```
(191,)
```

1.1 BUILDING LSTM MODEL

[169]:

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, LSTM, Dropout, InputLayer
```

[174]:

```
regressior = Sequential()



regressior.add(LSTM(units = 50, activation = 'relu', return_sequences = True,␣
  ↪input_shape = (x_train.shape[1], 5)))
regressior.add(Dropout(0.2))

regressior.add(LSTM(units = 60, activation = 'relu', return_sequences = True))
regressior.add(Dropout(0.3))

regressior.add(LSTM(units = 80, activation = 'relu', return_sequences = True))
regressior.add(Dropout(0.4))

regressior.add(LSTM(units = 120, activation = 'relu'))
regressior.add(Dropout(0.5))

regressior.add(Dense(units = 1))
```

[175]:

```
regressior.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_21 (LSTM)              (None, 60, 50)            11200

 dropout_19 (Dropout)        (None, 60, 50)            0

 lstm_22 (LSTM)              (None, 60, 60)            26640

 dropout_20 (Dropout)        (None, 60, 60)            0

 lstm_23 (LSTM)              (None, 60, 80)            45120

 dropout_21 (Dropout)        (None, 60, 80)            0

 lstm_24 (LSTM)              (None, 120)               96480

 dropout_22 (Dropout)        (None, 120)               0

 dense_4 (Dense)             (None, 1)                 121

=================================================================
Total params: 179561 (701.41 KB)
Trainable params: 179561 (701.41 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[176]:
```
regressior.compile(optimizer='adam', loss = 'mean_squared_error')
```

[177]:
```
regressior.fit(x_train, y_train, epochs=10, batch_size=32)
```

```
Epoch 1/10
6/6 [==============================] - 14s 260ms/step - loss: 0.2975
Epoch 2/10
6/6 [==============================] - 2s 261ms/step - loss: 0.0649
Epoch 3/10
6/6 [==============================] - 2s 270ms/step - loss: 0.0451
Epoch 4/10
6/6 [==============================] - 2s 270ms/step - loss: 0.0363
Epoch 5/10
6/6 [==============================] - 2s 264ms/step - loss: 0.0276
Epoch 6/10
6/6 [==============================] - 2s 268ms/step - loss: 0.0268
Epoch 7/10
6/6 [==============================] - 2s 255ms/step - loss: 0.0267
Epoch 8/10
6/6 [==============================] - 2s 258ms/step - loss: 0.0263
```

```
Epoch 9/10
6/6 [==============================] - 2s 259ms/step - loss: 0.0258
Epoch 10/10
6/6 [==============================] - 2s 256ms/step - loss: 0.0213
```

[177]:
```
<keras.src.callbacks.History at 0x1b140b2df10>
```

1.2 PREPARING TEST DATASET

[178]:
```python
data_test.head()
```

[178]:
```
         Date        Open        High         Low       Close   Adj Close  \
0  2022-10-03   97.220001   99.970001   97.019997   99.300003   99.300003
1  2022-10-04  101.040001  102.720001  101.040001  102.410004  102.410004
2  2022-10-05  100.690002  102.739998   99.739998  102.220001  102.220001
3  2022-10-06  101.500000  103.730003  101.500000  102.239998  102.239998
4  2022-10-07  100.650002  101.419998   99.209999   99.570000   99.570000

     Volume
0  24840000
1  22580900
2  18475500
3  17156200
4  24249900
```

[179]:
```python
data_training.tail(60)
```

[179]:
```
Empty DataFrame
Columns: [Date, Open, High, Low, Close, Adj Close, Volume]
Index: []
```

[180]:
```python
past_60_days = data_training.tail(60)
```

[182]:
```python
import pandas as pd
df = pd.concat([past_60_days, data_test], ignore_index = True)
df = df.drop(['Date', 'Adj Close'], axis = 1)
df.head()
```

[182]:
```
         Open        High         Low       Close    Volume
0   97.220001   99.970001   97.019997   99.300003  24840000
1  101.040001  102.720001  101.040001  102.410004  22580900
2  100.690002  102.739998   99.739998  102.220001  18475500
3  101.500000  103.730003  101.500000  102.239998  17156200
4  100.650002  101.419998   99.209999   99.570000  24249900
```

[183]:
```python
inputs = scaler.transform(df)
inputs
```

```
[183]: array([[0.21961739, 0.25140503, 0.25046139, 0.28486492, 0.1823608 ],
              [0.2912603 , 0.30292246, 0.32465857, 0.34090097, 0.15704331],
              [0.28469617, 0.30329708, 0.30066443, 0.33747749, 0.11103453],
              …,
              [0.84733683, 0.89228173, 0.87190842, 0.89441443, 0.10796272],
              [0.91091523, 0.90558271, 0.88353641, 0.87135139, 0.1642527 ],
              [0.8748124 , 0.91438754, 0.89728678, 0.93117105, 0.11903065]])
```

```
[184]: x_test = []
       y_test = []

       for i in range(60, inputs.shape[0]):
           x_test.append(inputs[i-60:i])
           y_test.append(inputs[i, 0])
```

```
[185]: x_test, y_test = np.array(x_test), np.array(y_test)
       x_test.shape, y_test.shape
```

```
[185]: ((191, 60, 5), (191,))
```

```
[186]: y_pred = regressior.predict(x_test)
```

```
6/6 [==============================] - 2s 83ms/step
```

```
[187]: scaler.scale_
```

```
[187]: array([1.87546887e-02, 1.87336116e-02, 1.84569925e-02, 1.80180157e-02,
              1.12068927e-08])
```

```
[189]: scale = 1/1.87546887e-02
       scale
```

```
[189]: 53.319999920873116
```

```
[190]: y_pred = y_pred*scale
       y_test = y_test*scale
```

```
[191]: y_pred
```

```
[191]: array([[13.686212],
              [13.391365],
              [13.075833],
              [12.747404],
              [12.414974],
              [12.086359],
              [11.766061],
              [11.457394],
              [11.163772],
              [10.888061],
              [10.633193],
              [10.403312],
              [10.202367],
              [10.033145],
              [ 9.897481],
              [ 9.795105],
              [ 9.727857],
              [ 9.701569],
              [ 9.723241],
              [ 9.792483],
              [ 9.908464],
              [10.070756],
              [10.276862],
              [10.522423],
              [10.803728],
              [11.122309],
              [11.478851],
              [11.87184 ],
              [12.299011],
              [12.744063],
              [13.177891],
              [13.574702],
              [13.91383 ],
              [14.181498],
              [14.37296 ],
              [14.49057 ],
              [14.54004 ],
              [14.527274],
              [14.457587],
              [14.336392],
              [14.168411],
              [13.959737],
              [13.717005],
              [13.447864],
              [13.161731],
              [12.869303],
              [12.58286 ],
              [12.313871],
              [12.071705],
              [11.861868],
              [11.683571],
              [11.532044],
              [11.404003],
              [11.297625],
              [11.214199],
              [11.156974],
              [11.133454],
              [11.15345 ],
              [11.227778],
              [11.366047],
              [11.57489 ],
              [11.854435],
              [12.196351],
              [12.588174],
              [13.015895],
              [13.46702 ],
              [13.932345],
              [14.406782],
              [14.886986],
              [15.370728],
              [15.857303],
              [16.345747],
              [16.833897],
              [17.31999 ],
              [17.80318 ],
              [18.27895 ],
              [18.742989],
              [19.189661],
              [19.614412],
              [20.014074],
              [20.387104],
              [20.731234],
              [21.042145],
              [21.317747],
              [21.560898],
              [21.777498],
              [21.972752],
              [22.151148],
              [22.315199],
              [22.46644 ],
              [22.606709],
              [22.738838],
              [22.865374],
              [22.993042],
              [23.133957],
              [23.301737],
              [23.507324],
              [23.760813],
              [24.071184],
              [24.445263],
              [24.8865  ],
              [25.394196],
              [25.961666],
              [26.579597],
              [27.238716],
              [27.929874],
              [28.639832],
              [29.357483],
              [30.075626],
              [30.78898 ],
              [31.495651],
              [32.190174],
              [32.863926],
              [33.50931 ],
              [34.120007],
              [34.693768],
              [35.22727 ],
              [35.72005 ],
              [36.168545],
              [36.572567],
              [36.932205],
              [37.24738 ],
              [37.51815 ],
              [37.74392 ],
              [37.921432],
              [38.04942 ],
              [38.130466],
              [38.1697  ],
              [38.17401 ],
              [38.151024],
              [38.106907],
              [38.046906],
              [37.97012 ],
              [37.874924],
              [37.762894],
              [37.63982 ],
              [37.5169  ],
              [37.407127],
              [37.31992 ],
              [37.260784],
              [37.22808 ],
              [37.21093 ],
              [37.203583],
              [37.202038],
              [37.21106 ],
              [37.24127 ],
              [37.306095],
              [37.419025],
              [37.588318],
              [37.81363 ],
              [38.09005 ],
              [38.410557],
              [38.76823 ],
              [39.156116],
              [39.568466],
              [39.999638],
              [40.441704],
              [40.887016],
              [41.33011 ],
              [41.764072],
              [42.182858],
              [42.57875 ],
              [42.944706],
              [43.27748 ],
              [43.57833 ],
              [43.852844],
              [44.10413 ],
              [44.336475],
              [44.5554  ],
              [44.769577],
              [44.988182],
              [45.221813],
              [45.47719 ],
              [45.757183],
              [46.060646],
              [46.384945],
              [46.72828 ],
              [47.08743 ],
              [47.458042],
              [47.836937],
              [48.21625 ],
              [48.592987],
              [48.966743],
              [49.33509 ],
              [49.68751 ],
              [50.011497],
              [50.296505],
              [50.531734],
              [50.710094],
              [50.829887],
              [50.89439 ]], dtype=float32)
```

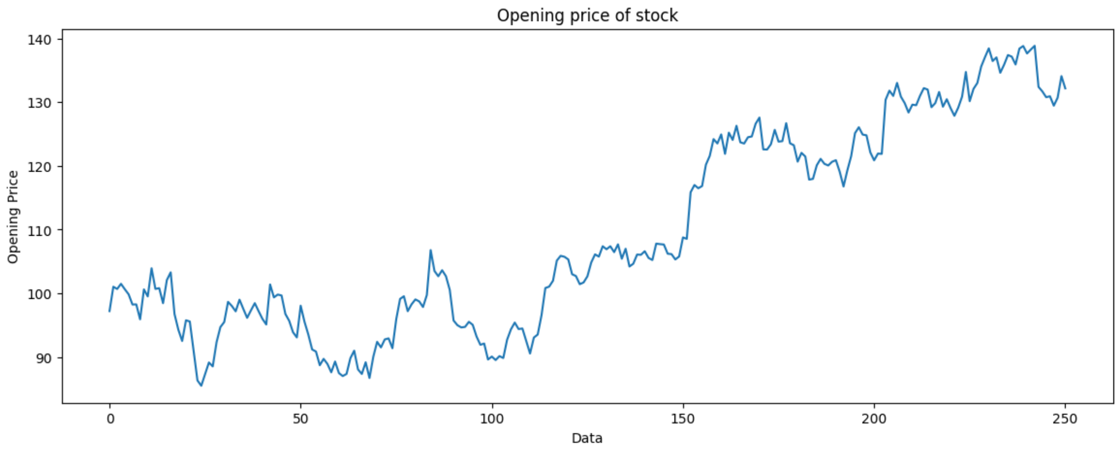
```
[192]: y_test
```

```
[192]: array([ 1.989998  ,  1.519997  ,  1.854996  ,  4.31999999,  5.49999999,
                2.559998  ,  1.849999  ,  3.68499799,  1.209999  ,  4.54999599,
                6.88999999,  6.01799799,  7.26999699,  7.42999999,  5.87999699,
10.43999498, 13.61999498, 14.04000098, 11.68999498, 12.76999698,
13.54000098, 13.23500098, 12.34999898, 14.22999598, 21.27999897,
17.99999997, 17.17499597, 18.11999497, 17.17999997, 15.02999898,
10.22999598,  9.49999999,  9.15000199,  9.22999599, 10.02999899,
 9.55999799,  7.72999599,  6.42399599,  6.61999499,  4.11999499,
 4.57999399,  4.02999899,  4.65000199,  4.34999899,  7.22999599,
 8.84999899,  9.90999599,  8.89499699,  8.97999599,  6.98999799,
 5.05499999,  7.55999799,  8.02999899, 11.05999798, 15.32999398,
15.54999598, 16.47000098, 19.62999697, 20.37999697, 20.22999597,
19.80999797, 17.48999797, 17.20999897, 15.92999998, 16.19999698,
17.15999597, 19.32999397, 20.61000097, 20.25999497, 21.87999697,
21.40999597, 21.87999697, 20.95999897, 22.17999997, 19.91999797,
21.48999797, 18.70499397, 19.13999997, 20.57999397, 20.54000097,
21.09999897, 20.04999597, 19.72000097, 22.29000097, 22.20999897,
22.15000197, 20.70999897, 20.65000197, 19.80999797, 20.28499597,
23.26999697, 23.04000097, 30.34999895, 31.48999795, 30.97999595,
31.31999995, 34.66999795, 36.04999595, 38.68999494, 37.99999994,
39.41999794, 36.36999495, 39.69999694, 38.55499994, 40.77999894,
38.18999494, 37.98999794, 38.97999594, 39.09999894, 41.08999594,
42.06499494, 37.07499694, 37.04999595, 37.88499494, 40.13999994,
38.29000094, 38.36999494, 41.18999494, 38.02500194, 37.72499894,
35.15000195, 36.52999895, 35.95600095, 32.32999395, 32.44999695,
34.57999395, 35.58999595, 34.80999795, 34.54999595, 35.12999695,
35.37999695, 33.55999795, 31.24999995, 33.79000095, 36.02999895,
39.61999494, 40.54999594, 39.39499694, 39.27999894, 36.61000095,
35.36000095, 36.41600095, 36.36999495, 44.84999893, 46.29000093,
45.45999893, 47.49999293, 45.34499393, 44.32999393, 42.85999294,
44.09000393, 43.99999293, 45.46999393, 46.67999993, 46.45999893,
43.69199394, 44.34000393, 46.07999393, 43.76999694, 44.93999493,
43.54999594, 42.33999594, 43.62000294, 45.34000393, 49.21700293,
44.62999693, 46.56999993, 47.48799893, 50.06000493, 51.54000092,
52.91999092, 50.92999992, 51.50499692, 49.09000393, 50.35999293,
51.87000292, 51.62000292, 50.38999193, 52.87999692, 53.29000092,
52.12000292, 52.73999792, 53.31999992, 46.87999693, 46.16999093,
45.26000193, 45.40399893, 43.92999993, 45.17999993, 48.56999993,
46.64499693])
```

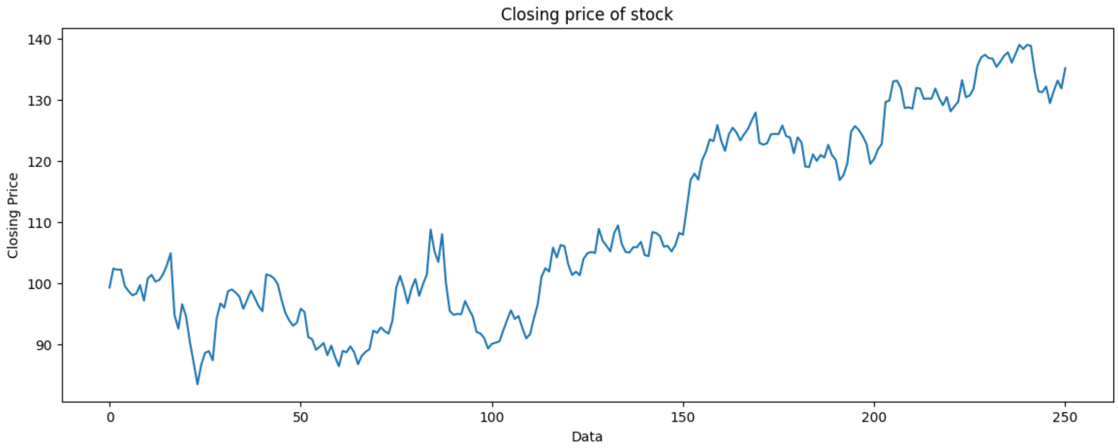
1.3 VISUALIZATION OF THE STOCK PRICE

[211]:

```
#visualizing the opening prices
plt.figure(figsize=(14,5))
plt.title('Opening price of stock')
plt.plot(df["Open"])
plt.xlabel('Data')
plt.ylabel('Opening Price')
plt.show()
```

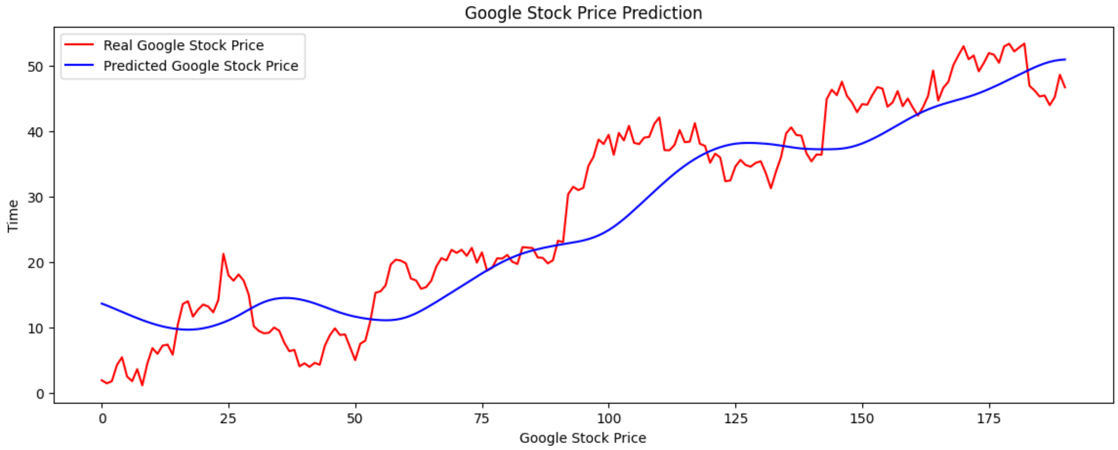


[210]:
```
#visualizing the closing prices
plt.figure(figsize=(14,5))
plt.title('Closing price of stock')
plt.plot(df["Close"])
plt.xlabel('Data')
plt.ylabel('Closing Price')
plt.show()
```



[209]:
```
#visualizing the real stock price and predicted stock price
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(14,5))
plt.plot(y_test, color='red', label= 'Real Google Stock Price')
plt.plot(y_pred, color='blue', label= 'Predicted Google Stock Price')
plt.title('Google Stock Price Prediction')
plt.xlabel('Google Stock Price')
plt.ylabel('Time')
plt.legend()
plt.show()
```



[ ]: