Deep Learning Lab Assignment 04

Name of Student: Aniket Yashvant Sandbhor
Branch: Computer Engineering
Exam number: B190424404

0.1 Implement Google stock price prediction using Recurrent Neural Network

[1]:
```
#import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
# read Dataset
df_train=pd.read_csv("Google_Stock_Price_Train.csv")
df_train.head(10)
```

[2]:
```
        Date    Open    High     Low   Close      Volume
0   1/3/2012  325.25  332.83  324.97  663.59   7,380,500
1   1/4/2012  331.27  333.87  329.08  666.45   5,749,400
2   1/5/2012  329.83  330.75  326.89  657.21   6,590,300
3   1/6/2012  328.34  328.77  323.68  648.24   5,405,900
4   1/9/2012  322.04  322.29  309.46  620.76  11,688,800
5  1/10/2012  313.70  315.72  307.30  621.43   8,824,000
6  1/11/2012  310.59  313.52  309.40  624.25   4,817,800
7  1/12/2012  314.43  315.26  312.08  627.92   3,764,400
8  1/13/2012  311.96  312.30  309.37  623.28   4,631,800
9  1/17/2012  314.81  314.81  311.67  626.86   3,832,800
```

[3]:
```
#keras only takes numpy array
#will use Open price for prediction so we need to make it NumPy array
training_set = df_train.iloc[:, 1: 2].values
training_set
```

```
[3]: array([[325.25],
            [331.27],
            [329.83],
            ...,
            [793.7 ],
            [783.33],
            [782.75]])
```

```
[4]: #scale the stock prices between (0, 1) to avoid intensive computation.
     from sklearn.preprocessing import MinMaxScaler
     sc= MinMaxScaler()
     training_set=sc.fit_transform(training_set)
     training_set
```

```
[4]: array([[0.08581368],
            [0.09701243],
            [0.09433366],
            ...,
            [0.95725128],
            [0.93796041],
            [0.93688146]])
```

```
[5]: x_train= training_set[0:1257]
     y_train= training_set[1:1258]
     display(x_train.shape,  y_train.shape)
```

```
(1257, 1)

(1257, 1)
```

```
[6]: x_train=np.reshape(x_train, (1257 , 1 , 1))
```

```
[7]: x_train.shape
```

```
[7]: (1257, 1, 1)
```

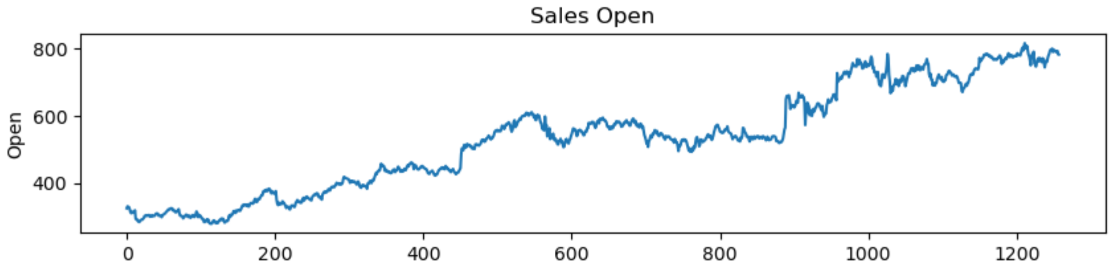
```
[8]: df_test=pd.read_csv("Google_Stock_Price_Test.csv")
     df_test
```

```
[8]:          Date    Open    High     Low   Close     Volume
     0    1/3/2017  778.81  789.63  775.80  786.14  1,657,300
     1    1/4/2017  788.36  791.34  783.16  786.90  1,073,000
     2    1/5/2017  786.08  794.48  785.02  794.02  1,335,200
     3    1/6/2017  795.26  807.90  792.20  806.15  1,640,200
     4    1/9/2017  806.40  809.97  802.83  806.65  1,272,400
     5   1/10/2017  807.86  809.13  803.51  804.79  1,176,800
     6   1/11/2017  805.00  808.15  801.37  807.91  1,065,900
     7   1/12/2017  807.14  807.39  799.17  806.36  1,353,100
     8   1/13/2017  807.48  811.22  806.69  807.88  1,099,200
```

```
9   1/17/2017  807.08  807.14  800.37  804.61  1,362,100
10  1/18/2017  805.81  806.21  800.99  806.07  1,294,400
11  1/19/2017  805.12  809.48  801.80  802.17    919,300
12  1/20/2017  806.91  806.91  801.69  805.02  1,670,000
13  1/23/2017  807.25  820.87  803.74  819.31  1,963,600
14  1/24/2017  822.30  825.90  817.82  823.87  1,474,000
15  1/25/2017  829.62  835.77  825.06  835.67  1,494,500
16  1/26/2017  837.81  838.00  827.01  832.15  2,973,900
17  1/27/2017  834.71  841.95  820.44  823.31  2,965,800
18  1/30/2017  814.66  815.84  799.80  802.32  3,246,600
19  1/31/2017  796.86  801.25  790.52  796.79  2,160,600
```

```
[9]: figure=plt.figure(figsize=(10,10))
     plt.subplots_adjust(top=1.35, bottom=1.2)
     df_train['Open'].plot()
     plt.ylabel('Open')
     plt.xlabel(None)
     plt.title(f"Sales Open")
```

```
[9]: Text(0.5, 1.0, 'Sales Open')
```



```
[10]: testing_set = df_test.iloc[:, 1: 2].values
      testing_set
```

```
[10]: array([[778.81],
             [788.36],
             [786.08],
             [795.26],
             [806.4 ],
             [807.86],
             [805.  ],
             [807.14],
             [807.48],
             [807.08],
             [805.81],
             [805.12],
             [806.91],
             [807.25],
             [822.3 ],
             [829.62],
             [837.81],
             [834.71],
             [814.66],
             [796.86]])
```

```
[11]: testing_set=sc.fit_transform(testing_set)
      testing_set.shape
```

```
[11]: (20, 1)
```

```
[12]: x_test= testing_set[0:20]
      y_test= testing_set[0:20]
      #display(x_test,  y_test)
      y_test.shape
```

```
[12]: (20, 1)
```

```
[13]: x_test=np.reshape(x_test, (20 , 1 , 1))
```

```
[14]: x_test.shape
```

```
[14]: (20, 1, 1)
```

```
[15]: import tensorflow.keras as tk
```

```
[16]: model = tk.Sequential()

      model.add(tk.layers.LSTM(units=5, activation= 'sigmoid', input_shape=
       ↪(None,1)))

      model.add(tk.layers.Dense( units=1 ))

      model.compile(optimizer='adam', loss='mean_squared_error')
      model.fit(x_train, y_train, batch_size=32,
       ↪epochs=50,validation_data=(x_test,y_test))
```

```
Epoch 1/50

2024-04-11 03:19:35.558716: W
tensorflow/tsl/platform/profile_utils/cpu_utils.cc:128] Failed to get
 ↪CPU
frequency: 0 Hz

40/40 [==============================] - 1s 3ms/step - loss: 0.9698 -
 ↪val_loss:
0.8649
```

```
Epoch 2/50
40/40 [==============================] - 0s 809us/step - loss: 0.7707 -
val_loss: 0.6841
Epoch 3/50
40/40 [==============================] - 0s 791us/step - loss: 0.6093 -
val_loss: 0.5390
Epoch 4/50
40/40 [==============================] - 0s 744us/step - loss: 0.4810 -
val_loss: 0.4225
Epoch 5/50
40/40 [==============================] - 0s 780us/step - loss: 0.3785 -
val_loss: 0.3300
Epoch 6/50
40/40 [==============================] - 0s 781us/step - loss: 0.2972 -
val_loss: 0.2571
Epoch 7/50
40/40 [==============================] - 0s 767us/step - loss: 0.2338 -
val_loss: 0.2002
Epoch 8/50
40/40 [==============================] - 0s 753us/step - loss: 0.1850 -
val_loss: 0.1564
Epoch 9/50
40/40 [==============================] - 0s 764us/step - loss: 0.1479 -
val_loss: 0.1236
Epoch 10/50
40/40 [==============================] - 0s 759us/step - loss: 0.1206 -
val_loss: 0.0995
Epoch 11/50
40/40 [==============================] - 0s 757us/step - loss: 0.1010 -
val_loss: 0.0825
Epoch 12/50
40/40 [==============================] - 0s 727us/step - loss: 0.0872 -
val_loss: 0.0707
Epoch 13/50
40/40 [==============================] - 0s 727us/step - loss: 0.0777 -
val_loss: 0.0625
Epoch 14/50
40/40 [==============================] - 0s 728us/step - loss: 0.0713 -
val_loss: 0.0569
Epoch 15/50
40/40 [==============================] - 0s 702us/step - loss: 0.0672 -
val_loss: 0.0534
Epoch 16/50
40/40 [==============================] - 0s 717us/step - loss: 0.0645 -
val_loss: 0.0510
Epoch 17/50
40/40 [==============================] - 0s 711us/step - loss: 0.0627 -
val_loss: 0.0493
```

```
Epoch 18/50
40/40 [==============================] - 0s 726us/step - loss: 0.0614 -
val_loss: 0.0482
Epoch 19/50
40/40 [==============================] - 0s 699us/step - loss: 0.0604 -
val_loss: 0.0473
Epoch 20/50
40/40 [==============================] - 0s 735us/step - loss: 0.0595 -
val_loss: 0.0465
Epoch 21/50
40/40 [==============================] - 0s 723us/step - loss: 0.0588 -
val_loss: 0.0459
Epoch 22/50
40/40 [==============================] - 0s 751us/step - loss: 0.0580 -
val_loss: 0.0453
Epoch 23/50
40/40 [==============================] - 0s 735us/step - loss: 0.0573 -
val_loss: 0.0446
Epoch 24/50
40/40 [==============================] - 0s 704us/step - loss: 0.0566 -
val_loss: 0.0441
Epoch 25/50
40/40 [==============================] - 0s 700us/step - loss: 0.0558 -
val_loss: 0.0435
Epoch 26/50
40/40 [==============================] - 0s 710us/step - loss: 0.0551 -
val_loss: 0.0429
Epoch 27/50
40/40 [==============================] - 0s 732us/step - loss: 0.0543 -
val_loss: 0.0423
Epoch 28/50
40/40 [==============================] - 0s 709us/step - loss: 0.0535 -
val_loss: 0.0417
Epoch 29/50
40/40 [==============================] - 0s 706us/step - loss: 0.0527 -
val_loss: 0.0410
Epoch 30/50
40/40 [==============================] - 0s 715us/step - loss: 0.0519 -
val_loss: 0.0404
Epoch 31/50
40/40 [==============================] - 0s 732us/step - loss: 0.0511 -
val_loss: 0.0397
Epoch 32/50
40/40 [==============================] - 0s 735us/step - loss: 0.0502 -
val_loss: 0.0391
Epoch 33/50
40/40 [==============================] - 0s 730us/step - loss: 0.0494 -
val_loss: 0.0384
```

```
Epoch 34/50
40/40 [==============================] - 0s 723us/step - loss: 0.0485 -
val_loss: 0.0377
Epoch 35/50
40/40 [==============================] - 0s 752us/step - loss: 0.0476 -
val_loss: 0.0370
Epoch 36/50
40/40 [==============================] - 0s 751us/step - loss: 0.0468 -
val_loss: 0.0364
Epoch 37/50
40/40 [==============================] - 0s 740us/step - loss: 0.0458 -
val_loss: 0.0356
Epoch 38/50
40/40 [==============================] - 0s 722us/step - loss: 0.0449 -
val_loss: 0.0349
Epoch 39/50
40/40 [==============================] - 0s 744us/step - loss: 0.0440 -
val_loss: 0.0342
Epoch 40/50
40/40 [==============================] - 0s 739us/step - loss: 0.0430 -
val_loss: 0.0334
Epoch 41/50
40/40 [==============================] - 0s 750us/step - loss: 0.0420 -
val_loss: 0.0327
Epoch 42/50
40/40 [==============================] - 0s 1ms/step - loss: 0.0410 -
 ↪val_loss:
0.0319
Epoch 43/50
40/40 [==============================] - 0s 769us/step - loss: 0.0400 -
val_loss: 0.0311
Epoch 44/50
40/40 [==============================] - 0s 759us/step - loss: 0.0390 -
val_loss: 0.0303
Epoch 45/50
40/40 [==============================] - 0s 735us/step - loss: 0.0380 -
val_loss: 0.0295
Epoch 46/50
40/40 [==============================] - 0s 737us/step - loss: 0.0369 -
val_loss: 0.0287
Epoch 47/50
40/40 [==============================] - 0s 742us/step - loss: 0.0358 -
val_loss: 0.0278
Epoch 48/50
40/40 [==============================] - 0s 740us/step - loss: 0.0347 -
val_loss: 0.0270
Epoch 49/50
```

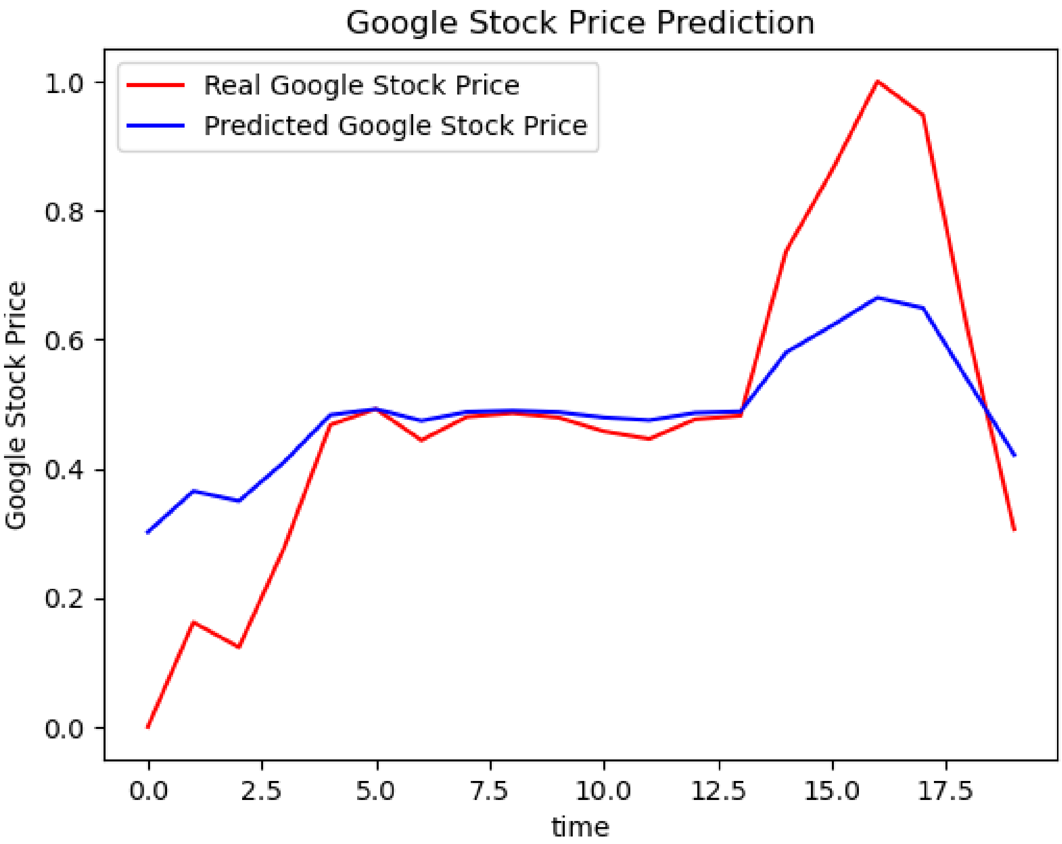
```
40/40 [==============================] - 0s 741us/step - loss: 0.0337 -
val_loss: 0.0261
Epoch 50/50
40/40 [==============================] - 0s 719us/step - loss: 0.0325 -
val_loss: 0.0253
```

```
[16]: <keras.callbacks.History at 0x179b2a5d0>
```

```
[17]: y_pred=model.predict(x_test)
```

```
1/1 [==============================] - 0s 70ms/step
```

```
[18]: 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( 'time' )
      plt.ylabel( 'Google Stock Price' )
      plt.legend()
      plt.show()
```


Google Stock Price Prediction
Real Google Stock Price
Predicted Google Stock Price
Google Stock Price
time