Submitted by Navneet Das 3433 Comp A

```
***'
Assignment 4 Deep Learning
Recurrent neural network (RNN) Use the Google stock prices dataset and
design a time series analysis and prediction system using RNN
'***
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
```

In [2]:
```
base = "DeepLearningData/Recurrent Neural Network"
```

In [3]:
```
train_data = pd.read_csv(base+"/Stock_Price_Train.csv")
```

In [4]:
```
train_data.head()
```

Out[4]:

	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

In [5]:
```
train = train_data.loc[:,['Open']].values
```

In [6]:
```
train
```

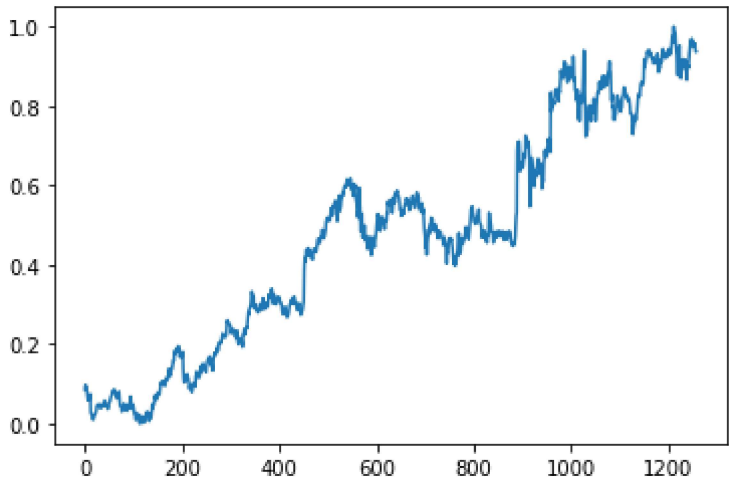
Out[6]:
```
array([[325.25],
       [331.27],
       [329.83],
       ...,
       [793.7 ],
       [783.33],
       [782.75]])
```

In [7]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0,1))
train_scaled = scaler.fit_transform(train)
train_scaled
```

Out[7]:
```
array([[0.08581368],
       [0.09701243],
       [0.09433366],
       ...,
       [0.95725128],
       [0.93796041],
       [0.93688146]])
```

In [8]:
```
plt.plot(train_scaled)
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x26c9daaedc0>]
```



In [9]:
```
X_train = []
y_train = []
timesteps = 50
for i in range(timesteps, 1258):
    X_train.append(train_scaled[i-timesteps:i, 0])
    y_train.append(train_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

In [10]:
```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train
```

Out[10]:
```
array([[[0.08581368],
        [0.09701243],
        [0.09433366],
        ...,
        [0.03675869],
        [0.04486941],
        [0.05065481]],

       [[0.09701243],
        [0.09433366],
        [0.09156187],
        ...,
        [0.04486941],
        [0.05065481],
        [0.05214302]],

       [[0.09433366],
        [0.09156187],
        [0.07984225],
        ...,
        [0.05065481],
        [0.05214302],
        [0.05612397]],

       ...,

       [[0.9313937 ],
        [0.94636878],
        [0.96569685],
        ...,
        [0.95475854],
        [0.95204256],
        [0.95163331]],

       [[0.94636878],
        [0.96569685],
        [0.97510976],
        ...,
        [0.95204256],
        [0.95163331],
        [0.95725128]],

       [[0.96569685],
        [0.97510976],
        [0.95966962],
        ...,
        [0.95163331],
        [0.95725128],
        [0.93796041]]])
```

In [11]:
```
y_train
```

Out[11]:
```
array([0.05214302, 0.05612397, 0.05818885, ..., 0.95725128, 0.93796041,
       0.93688146])
```

Create the RNN Model

In [12]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import SimpleRNN
from keras.layers import Dropout

regressor = Sequential()

regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True,
regressor.add(Dropout(0.2))

regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True)
regressor.add(Dropout(0.2))

regressor.add(SimpleRNN(units = 50,activation='tanh', return_sequences = True)
regressor.add(Dropout(0.2))

regressor.add(SimpleRNN(units = 50))
regressor.add(Dropout(0.2))

regressor.add(Dense(units = 1))

regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

```
38/38 [==============================] - 1s 20ms/step - loss: 0.0020
Epoch 93/100
38/38 [==============================] - 1s 24ms/step - loss: 0.0022
Epoch 94/100
38/38 [==============================] - 1s 23ms/step - loss: 0.0022
Epoch 95/100
38/38 [==============================] - 1s 25ms/step - loss: 0.0021
Epoch 96/100
38/38 [==============================] - 1s 26ms/step - loss: 0.0022
Epoch 97/100
38/38 [==============================] - 1s 23ms/step - loss: 0.0020
Epoch 98/100
38/38 [==============================] - 1s 22ms/step - loss: 0.0023
Epoch 99/100
38/38 [==============================] - 1s 21ms/step - loss: 0.0019
Epoch 100/100

38/38 [==============================] - 1s 21ms/step - loss: 0.0021
```

Out[12]:
```
<keras.callbacks.History at 0x26ca836f430>
```

In [13]:
```
test_data = pd.read_csv(base+'/Stock_Price_Test.csv')
```

In [14]:
```
test_data.head()
```

Out[14]:

	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

In [15]:
```
real_stock_price = test_data.loc[:,['Open']].values
```

In [16]:
```
real_stock_price
```

Out[16]:
```
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]])
```

In [17]:
```
total_data = pd.concat((train_data['Open'],test_data['Open']),axis=0)
inputs = total_data[len(total_data)-len(test_data)-timesteps:].values.reshape(
inputs = scaler.transform(inputs) #min max scaler
```

In [18]:
```
inputs
```

Out[18]:
```
array([[0.97510976],
       [0.95966962],
       [0.97808617],
       [1.        ],
       [0.98076494],
       [0.97083116],
       [0.98450406],
       [0.96054394],
       [0.9371419 ],
       [0.92841729],
       [0.90804747],
       [0.8771858 ],
       [0.92153434],
       [0.93809063],
       [0.93165414],
       [0.95254483],
       [0.88812412],
       [0.88637547],
       [0.87032145],
       [0.88563137],
       [0.90743359],
       [0.91571173],
       [0.89941588],
       [0.91805566],
       [0.9089404 ],
       [0.9024853 ],
       [0.89456061],
       [0.91600938],
       [0.9132934 ],
       [0.88979835],
       [0.86589404],
       [0.89030062],
       [0.90335962],
       [0.89642086],
       [0.91777662],
       [0.93176576],
       [0.94114145],
       [0.95762334],
       [0.96413424],
       [0.96402262],
       [0.96971501],
       [0.95077759],
       [0.96294367],
       [0.96123223],
       [0.95475854],
       [0.95204256],
       [0.95163331],
       [0.95725128],
       [0.93796041],
       [0.93688146],
       [0.92955205],
       [0.94731751],
       [0.94307612],
       [0.96015329],
       [0.98087655],
       [0.98359253],
       [0.97827219],
```

```
[0.98225314],
[0.98288563],
[0.98214153],
[0.979779  ],
[0.97849542],
[0.98182528],
[0.98245777],
[1.01045465],
[1.02407173],
[1.03930724],
[1.03354044],
[0.99624228],
[0.9631297 ]])
```

In [19]:
```
X_test = []
for i in range(timesteps, 70):
    X_test.append(inputs[i-timesteps:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = scaler.inverse_transform(predicted_stock_price)
```

```
1/1 [==============================] - 0s 398ms/step
```

Visualization

In [20]:
```
plt.plot(real_stock_price,color='red',label='Real Google Stock Price')
plt.plot(predicted_stock_price,color='blue',label='Predicted Google Stock Pric
plt.title('Google Stoc Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```