# Recurrent Neural Network

## Part 1 - Data Preprocessing

### Importing the libraries

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

### Importing the training set

In [2]:
```
dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')
dataset_train.head()
```

Out[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 |

In [4]:
```
training_set = dataset_train.iloc[:, 1:2].values
training_set
```

Out[4]:
```
array([[325.25],
       [331.27],
       [329.83],
       ...,
       [793.7 ],
       [783.33],
       [782.75]])
```

### Feature Scaling

In [5]:
```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
training_set_scaled = sc.fit_transform(training_set)
```

In [6]:
```
training_set.shape
```

Out[6]:
```
(1258, 1)
```

### Creating a data structure with 60 timesteps and 1 output

In [7]:
```
X_train = []
y_train = []
for i in range(60, 1258):
    X_train.append(training_set_scaled[i-60:i, 0])
    y_train.append(training_set_scaled[i, 0])


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

### Reshaping

In [8]:
```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

## Part 2 - Building and Training the RNN

### Importing the Keras libraries and packages

In [9]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

### Initialising the RNN

In [10]:
```
regressor = Sequential()
```

### Adding the first LSTM layer and some Dropout regularisation

In [11]:
```
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train
.shape[1], 1)))
regressor.add(Dropout(0.2))
```

### Adding a second LSTM layer and some Dropout regularisation

In [12]:
```
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
```

### Adding a third LSTM layer and some Dropout regularisation

In [13]:
```
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))
```

### Adding a fourth LSTM layer and some Dropout regularisation

In [14]:
```
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))
```

### Adding the output layer

In [15]:
```
regressor.add(Dense(units = 1))
```

### Compiling the RNN

In [16]:
```
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

### Fitting the RNN to the Training set

In [17]:
```
regressor.fit(X_train, y_train, epochs = 100, batch_size = 32)
```

```
Epoch 1/100
38/38 [==============================] - 32s 153ms/step - loss: 0.0983
Epoch 2/100
38/38 [==============================] - 5s 143ms/step - loss: 0.0077
Epoch 3/100
38/38 [==============================] - 8s 218ms/step - loss: 0.0058
Epoch 4/100
38/38 [==============================] - 7s 188ms/step - loss: 0.0048
Epoch 5/100
38/38 [==============================] - 5s 140ms/step - loss: 0.0047
Epoch 6/100
38/38 [==============================] - 6s 146ms/step - loss: 0.0050
Epoch 7/100
38/38 [==============================] - 8s 209ms/step - loss: 0.0051
Epoch 8/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0053
Epoch 9/100
38/38 [==============================] - 5s 143ms/step - loss: 0.0048
Epoch 10/100
38/38 [==============================] - 5s 140ms/step - loss: 0.0042
Epoch 11/100
38/38 [==============================] - 8s 199ms/step - loss: 0.0048
Epoch 12/100
38/38 [==============================] - 8s 200ms/step - loss: 0.0037
Epoch 13/100
38/38 [==============================] - 6s 165ms/step - loss: 0.0042
Epoch 14/100
38/38 [==============================] - 7s 181ms/step - loss: 0.0041
Epoch 15/100
38/38 [==============================] - 6s 167ms/step - loss: 0.0038
Epoch 16/100
38/38 [==============================] - 6s 156ms/step - loss: 0.0047
Epoch 17/100
38/38 [==============================] - 7s 184ms/step - loss: 0.0033
Epoch 18/100
38/38 [==============================] - 8s 202ms/step - loss: 0.0039
Epoch 19/100
38/38 [==============================] - 8s 197ms/step - loss: 0.0037
Epoch 20/100
38/38 [==============================] - 6s 145ms/step - loss: 0.0034
Epoch 21/100
38/38 [==============================] - 6s 170ms/step - loss: 0.0034
Epoch 22/100
38/38 [==============================] - 8s 207ms/step - loss: 0.0035
Epoch 23/100
38/38 [==============================] - 6s 161ms/step - loss: 0.0034
Epoch 24/100
38/38 [==============================] - 8s 198ms/step - loss: 0.0038
Epoch 25/100
38/38 [==============================] - 8s 212ms/step - loss: 0.0034
Epoch 26/100
38/38 [==============================] - 6s 154ms/step - loss: 0.0040
Epoch 27/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0033
Epoch 28/100
38/38 [==============================] - 6s 161ms/step - loss: 0.0029
Epoch 29/100
```

```
38/38 [==============================] - 9s 244ms/step - loss: 0.0039
Epoch 30/100
38/38 [==============================] - 7s 177ms/step - loss: 0.0033
Epoch 31/100
38/38 [==============================] - 6s 158ms/step - loss: 0.0029
Epoch 32/100
38/38 [==============================] - 7s 191ms/step - loss: 0.0032
Epoch 33/100
38/38 [==============================] - 6s 169ms/step - loss: 0.0035
Epoch 34/100
38/38 [==============================] - 7s 186ms/step - loss: 0.0032
Epoch 35/100
38/38 [==============================] - 9s 223ms/step - loss: 0.0033 0s - lo
ss: 0.003
Epoch 36/100
38/38 [==============================] - 9s 243ms/step - loss: 0.0031
Epoch 37/100
38/38 [==============================] - 6s 153ms/step - loss: 0.0030
Epoch 38/100
38/38 [==============================] - 6s 164ms/step - loss: 0.0031
Epoch 39/100
38/38 [==============================] - 7s 194ms/step - loss: 0.0025
Epoch 40/100
38/38 [==============================] - 6s 151ms/step - loss: 0.0035
Epoch 41/100
38/38 [==============================] - 6s 151ms/step - loss: 0.0025
Epoch 42/100
38/38 [==============================] - 7s 196ms/step - loss: 0.0024
Epoch 43/100
38/38 [==============================] - 9s 237ms/step - loss: 0.0025
Epoch 44/100
38/38 [==============================] - 6s 168ms/step - loss: 0.0030
Epoch 45/100
38/38 [==============================] - 7s 198ms/step - loss: 0.0023
Epoch 46/100
38/38 [==============================] - 8s 196ms/step - loss: 0.0026
Epoch 47/100
38/38 [==============================] - 6s 167ms/step - loss: 0.0032
Epoch 48/100
38/38 [==============================] - 6s 162ms/step - loss: 0.0024
Epoch 49/100
38/38 [==============================] - 11s 280ms/step - loss: 0.0024
Epoch 50/100
38/38 [==============================] - 7s 181ms/step - loss: 0.0023
Epoch 51/100
38/38 [==============================] - 6s 146ms/step - loss: 0.0027
Epoch 52/100
38/38 [==============================] - 7s 178ms/step - loss: 0.0022
Epoch 53/100
38/38 [==============================] - 7s 176ms/step - loss: 0.0023
Epoch 54/100
38/38 [==============================] - 7s 179ms/step - loss: 0.0022
Epoch 55/100
38/38 [==============================] - 6s 161ms/step - loss: 0.0021
Epoch 56/100
38/38 [==============================] - 9s 234ms/step - loss: 0.0023
Epoch 57/100
```

```
38/38 [==============================] - 6s 158ms/step - loss: 0.0021
Epoch 58/100
38/38 [==============================] - 5s 144ms/step - loss: 0.0025
Epoch 59/100
38/38 [==============================] - 6s 166ms/step - loss: 0.0020 1s - lo
s - ETA: 0s - loss: 0.0
Epoch 60/100
38/38 [==============================] - 7s 174ms/step - loss: 0.0019
Epoch 61/100
38/38 [==============================] - 6s 172ms/step - loss: 0.0021
Epoch 62/100
38/38 [==============================] - 6s 157ms/step - loss: 0.0022
Epoch 63/100
38/38 [==============================] - 8s 202ms/step - loss: 0.0020 2s -
Epoch 64/100
38/38 [==============================] - 8s 195ms/step - loss: 0.0019
Epoch 65/100
38/38 [==============================] - 6s 150ms/step - loss: 0.0021
Epoch 66/100
38/38 [==============================] - 6s 157ms/step - loss: 0.0020
Epoch 67/100
38/38 [==============================] - 7s 186ms/step - loss: 0.0021
Epoch 68/100
38/38 [==============================] - 6s 145ms/step - loss: 0.0022
Epoch 69/100
38/38 [==============================] - 6s 147ms/step - loss: 0.0020
Epoch 70/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0017 0s - lo
ss: 0.001
Epoch 71/100
38/38 [==============================] - 9s 243ms/step - loss: 0.0017
Epoch 72/100
38/38 [==============================] - 7s 175ms/step - loss: 0.0016
Epoch 73/100
38/38 [==============================] - 5s 144ms/step - loss: 0.0017
Epoch 74/100
38/38 [==============================] - 7s 178ms/step - loss: 0.0018
Epoch 75/100
38/38 [==============================] - 6s 168ms/step - loss: 0.0018
Epoch 76/100
38/38 [==============================] - 5s 140ms/step - loss: 0.0018
Epoch 77/100
38/38 [==============================] - 6s 144ms/step - loss: 0.0017
Epoch 78/100
38/38 [==============================] - 6s 169ms/step - loss: 0.0015
Epoch 79/100
38/38 [==============================] - 8s 205ms/step - loss: 0.0019
Epoch 80/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0018
Epoch 81/100
38/38 [==============================] - 6s 150ms/step - loss: 0.0017
Epoch 82/100
38/38 [==============================] - 7s 189ms/step - loss: 0.0016
Epoch 83/100
38/38 [==============================] - 6s 149ms/step - loss: 0.0016
Epoch 84/100
38/38 [==============================] - 6s 158ms/step - loss: 0.0017
```

```
Epoch 85/100
38/38 [==============================] - 6s 168ms/step - loss: 0.0019
Epoch 86/100
38/38 [==============================] - 8s 205ms/step - loss: 0.0015
Epoch 87/100
38/38 [==============================] - 7s 172ms/step - loss: 0.0017TA: 1s -
l
Epoch 88/100
38/38 [==============================] - 5s 144ms/step - loss: 0.0016
Epoch 89/100
38/38 [==============================] - 8s 203ms/step - loss: 0.0015
Epoch 90/100
38/38 [==============================] - 7s 175ms/step - loss: 0.0016
Epoch 91/100
38/38 [==============================] - 6s 149ms/step - loss: 0.0017
Epoch 92/100
38/38 [==============================] - 6s 147ms/step - loss: 0.0015
Epoch 93/100
38/38 [==============================] - 7s 183ms/step - loss: 0.0017
Epoch 94/100
38/38 [==============================] - 8s 200ms/step - loss: 0.0014
Epoch 95/100
38/38 [==============================] - 6s 145ms/step - loss: 0.0015
Epoch 96/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0014
Epoch 97/100
38/38 [==============================] - 7s 187ms/step - loss: 0.0015
Epoch 98/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0016
Epoch 99/100
38/38 [==============================] - 5s 136ms/step - loss: 0.0016
Epoch 100/100
38/38 [==============================] - 5s 142ms/step - loss: 0.0014
```

Out[17]: `<tensorflow.python.keras.callbacks.History at 0x25c88da8bb0>`

In [18]:
```
pred=regressor.predict(X_train)
```

### Evaluating the RNN

In [19]:
```
import math
from sklearn.metrics import mean_squared_error
rmse = math.sqrt(mean_squared_error(y_train,pred))
rmse
```

Out[19]: `0.027069358807676185`

## Part 3 - Making the predictions and visualising the results

## Getting the real stock price of 2017

In [20]:
```
dataset_test = pd.read_csv('Google_Stock_Price_Test.csv')
real_stock_price = dataset_test.iloc[:, 1:2].values
```

## Getting the predicted stock price of 2017

In [21]:
```
dataset_total = pd.concat((dataset_train['Open'], dataset_test['Open']), axis
= 0)
inputs = dataset_total[len(dataset_total) - len(dataset_test) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, 80):
    X_test.append(inputs[i-60: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 = sc.inverse_transform(predicted_stock_price)
```

## Visualising the results

In [22]:
```
plt.plot(real_stock_price, color = 'red', label = 'Real Google Stock Price')
plt.plot(predicted_stock_price, color = 'blue', label = 'Predicted Google Stoc
k Price')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```