```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
train_data = pd.read_csv("Google_Stock_Price_Train.csv")
train_data.head()
```

	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

```
train_data_set = train_data.iloc[:,1:2].values
```

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
train_data_scaled = sc.fit_transform(train_data_set)
```

```
X_train = []
y_train = []
for i in range(60, len(train_data_set)):
  X_train.append(train_data_scaled[i-60:i, 0])
  y_train.append(train_data_scaled[i,0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

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

Build a RNN Model

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(Dropout(0.3))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(units=50, return_sequences=True))
model.add(Dropout(0.3))
model.add(LSTM(units=50, return_sequences=False))  # Last LSTM layer, no need for return_sequences=True
model.add(Dropout(0.3))
```

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

```
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

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

```
Epoch 1/100
38/38 [==============================] - 18s 181ms/step - loss: 0.0428
Epoch 2/100
38/38 [==============================] - 6s 158ms/step - loss: 0.0095
Epoch 3/100
38/38 [==============================] - 5s 142ms/step - loss: 0.0068
Epoch 4/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0069
```

```
Epoch 5/100
38/38 [==============================] - 8s 204ms/step - loss: 0.0064
Epoch 6/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0065
Epoch 7/100
38/38 [==============================] - 8s 208ms/step - loss: 0.0060
Epoch 8/100
38/38 [==============================] - 5s 137ms/step - loss: 0.0064
Epoch 9/100
38/38 [==============================] - 7s 173ms/step - loss: 0.0060
Epoch 10/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0050
Epoch 11/100
38/38 [==============================] - 6s 169ms/step - loss: 0.0056
Epoch 12/100
38/38 [==============================] - 5s 133ms/step - loss: 0.0050
Epoch 13/100
38/38 [==============================] - 5s 138ms/step - loss: 0.0056
Epoch 14/100
38/38 [==============================] - 6s 165ms/step - loss: 0.0052
Epoch 15/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0048
Epoch 16/100
38/38 [==============================] - 6s 169ms/step - loss: 0.0053
Epoch 17/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0043
Epoch 18/100
38/38 [==============================] - 6s 169ms/step - loss: 0.0043
Epoch 19/100
38/38 [==============================] - 5s 134ms/step - loss: 0.0043
Epoch 20/100
38/38 [==============================] - 6s 145ms/step - loss: 0.0044
Epoch 21/100
38/38 [==============================] - 6s 155ms/step - loss: 0.0040
Epoch 22/100
38/38 [==============================] - 5s 133ms/step - loss: 0.0041
Epoch 23/100
38/38 [==============================] - 6s 168ms/step - loss: 0.0040
Epoch 24/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0036
Epoch 25/100
38/38 [==============================] - 6s 169ms/step - loss: 0.0039
Epoch 26/100
38/38 [==============================] - 5s 133ms/step - loss: 0.0039
Epoch 27/100
38/38 [==============================] - 6s 152ms/step - loss: 0.0041
Epoch 28/100
38/38 [==============================] - 6s 151ms/step - loss: 0.0041
Epoch 29/100
38/38 [==============================] - 5s 135ms/step - loss: 0.0042
```

```
test_data = pd.read_csv('Google_Stock_Price_Test.csv')
test_data_price = test_data.iloc[:, 1:2].values


dataset_total = pd.concat((train_data['Open'], test_data['Open']), axis = 0)
inputs = dataset_total[len(dataset_total) - len(test_data) - 60:].values
inputs = inputs.reshape(-1,1)
inputs = sc.transform(inputs)
X_test = []
for i in range(60, len(inputs)):
    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 = model.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

```
1/1 [==============================] - 3s 3s/step
```

```
plt.plot(test_data_price, color = 'red', label = 'Real Google Stock Price')
plt.plot(predicted_stock_price, 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()
```