```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
dataset_train=pd.read_csv('Google_Stock_Price_Train.csv')
print(dataset_train)
training_set=dataset_train.iloc[:,1:2].values
print(training_set)
```

```
            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
...          ...     ...     ...     ...     ...         ...
1253  12/23/2016  790.90  792.74  787.28  789.91     623,400
1254  12/27/2016  790.68  797.86  787.66  791.55     789,100
1255  12/28/2016  793.70  794.23  783.20  785.05   1,153,800
1256  12/29/2016  783.33  785.93  778.92  782.79     744,300
1257  12/30/2016  782.75  782.78  770.41  771.82   1,770,000

[1258 rows x 6 columns]
[[325.25]
 [331.27]
 [329.83]
 ...
 [793.7 ]
 [783.33]
 [782.75]]
```

```
#Perform the feature Scalling
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler(feature_range=(0,1))
training_set_scaled=sc.fit_transform(training_set)
print(training_set_scaled)
```

```
[[0.08581368]
 [0.09701243]
 [0.09433366]
 ...
 [0.95725128]
 [0.93796041]
 [0.93688146]]
```

```
x_train=[]
y_train=[]

for i in range(60,len(training_set_scaled)):
    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)
print(x_train)
x_train=np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
print(x_train)
```

```
[[0.08581368 0.09701243 0.09433366 ... 0.07846566 0.08034452
0.08497656]
 [0.09701243 0.09433366 0.09156187 ... 0.08034452 0.08497656
0.08627874]
 [0.09433366 0.09156187 0.07984225 ... 0.08497656 0.08627874
0.08471612]
 ...
 [0.92106928 0.92438053 0.93048218 ... 0.95475854 0.95204256
0.95163331]
 [0.92438053 0.93048218 0.9299055  ... 0.95204256 0.95163331
0.95725128]
 [0.93048218 0.9299055  0.93113327 ... 0.95163331 0.95725128
0.93796041]]
[[[0.08581368]
  [0.09701243]
  [0.09433366]
  ...
  [0.07846566]
  [0.08034452]
  [0.08497656]]

 [[0.09701243]
  [0.09433366]
  [0.09156187]
  ...
  [0.08034452]
  [0.08497656]
  [0.08627874]]

 [[0.09433366]
  [0.09156187]
  [0.07984225]
  ...
  [0.08497656]
  [0.08627874]
  [0.08471612]]

 ...

 [[0.92106928]
  [0.92438053]
  [0.93048218]
  ...
  [0.95475854]
  [0.95204256]
```

```
  [0.95163331]]

 [[0.92438053]
  [0.93048218]
  [0.9299055 ]
  ...
  [0.95204256]
  [0.95163331]
  [0.95725128]]

 [[0.93048218]
  [0.9299055 ]
  [0.93113327]
  ...
  [0.95163331]
  [0.95725128]
  [0.93796041]]]
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
model=Sequential()
#Add LSTM layer
model.add(LSTM(units=50,return_sequences=True,input_shape=(x_train.sha
pe[1],1)))
#add regularization
model.add(Dropout(0.2))
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50,return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=50))
model.add(Dropout(0.2))
#add output layer
model.add(Dense(units=1))
model.summary()

model.compile(optimizer='adam',loss='mean_squared_error')
model.fit(x_train,y_train,epochs=40,batch_size=32)
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_21 (LSTM)               (None, 60, 50)            10400

 dropout_21 (Dropout)         (None, 60, 50)            0

 lstm_22 (LSTM)               (None, 60, 50)            20200
```

```
 dropout_22 (Dropout)        (None, 60, 50)            0

 lstm_23 (LSTM)              (None, 60, 50)            20200

 dropout_23 (Dropout)        (None, 60, 50)            0

 lstm_24 (LSTM)              (None, 50)                20200

 dropout_24 (Dropout)        (None, 50)                0

 dense_3 (Dense)             (None, 1)                 51

=================================================================
Total params: 71051 (277.54 KB)
Trainable params: 71051 (277.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/40
38/38 [==============================] - 4s 29ms/step - loss: 0.0445
Epoch 2/40
38/38 [==============================] - 1s 31ms/step - loss: 0.0072
Epoch 3/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0059
Epoch 4/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0071
Epoch 5/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0055
Epoch 6/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0048
Epoch 7/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0050
Epoch 8/40
38/38 [==============================] - 1s 31ms/step - loss: 0.0050
Epoch 9/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0050
Epoch 10/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0046
Epoch 11/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0040
Epoch 12/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0048
Epoch 13/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0036
Epoch 14/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0042
Epoch 15/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0037
Epoch 16/40
```

```
38/38 [==============================] - 1s 29ms/step - loss: 0.0036
Epoch 17/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0039
Epoch 18/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0037
Epoch 19/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0035
Epoch 20/40
38/38 [==============================] - 1s 30ms/step - loss: 0.0035
Epoch 21/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0040
Epoch 22/40
38/38 [==============================] - 1s 30ms/step - loss: 0.0035
Epoch 23/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0033
Epoch 24/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0037
Epoch 25/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0032
Epoch 26/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0034
Epoch 27/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0032
Epoch 28/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0029
Epoch 29/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0033
Epoch 30/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0031
Epoch 31/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0027
Epoch 32/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0029
Epoch 33/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0032
Epoch 34/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0032
Epoch 35/40
38/38 [==============================] - 1s 30ms/step - loss: 0.0028
Epoch 36/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0027
Epoch 37/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0029
Epoch 38/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0026
Epoch 39/40
38/38 [==============================] - 1s 28ms/step - loss: 0.0029
Epoch 40/40
38/38 [==============================] - 1s 29ms/step - loss: 0.0029
```

```
<keras.src.callbacks.History at 0x7f3baf52ae50>
```

```
test_df=pd.read_csv('Google_Stock_Price_Test.xls')
test_df.head()
stock_price=test_df.iloc[:,1:2].values
stock_price
```

```
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]])
```

```
#fetch 60 timesteps by combining train and test
total_df=pd.concat((dataset_train['Open'],test_df['Open']),axis=0)
inputs=total_df[0:].values
inputs=inputs.reshape(-1,1)
inputs=sc.transform(inputs)
#reshape the dataset
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)
print(predicted_stock_price)
```

```
39/39 [==============================] - 1s 9ms/step
[[0.07819058]
 [0.08196405]
 [0.08571862]
 ...
 [0.9666085 ]
 [0.9712523 ]
 [0.9759196 ]]
```

```
predicted_stock_price=sc.inverse_transform(predicted_stock_price)
predicted_stock_price
```

```
array([[9.2953896e+07],
       [9.3540048e+07],
       [9.4123296e+07],
       ...,
       [2.3095995e+08],
       [2.3168133e+08],
       [2.3240632e+08]], dtype=float32)
```

```
plt.plot(stock_price,color='red',label='Real Google Stock price')
plt.plot(predicted_stock_price,color='green',label='Predicted Google
Stock price')
plt.title('Google Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock price')
plt.legend()
plt.show()
```