Stock Price Predicition

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

Data Preprocessing

```
#loading the Data
dataset_train = pd.read_csv('Google_Stock_Price_Train.csv')
print('shape is = {}'.format(dataset_train.shape))
print(dataset_train.head())
```

```
shape is = (1258, 6)
       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
```

```
training_set = dataset_train.iloc[:,1:2].values
print('shape is ={}'.format(training_set.shape))
print(training_set[0:5])
```

```
shape is =(1258, 1)
[[325.25]
 [331.27]
 [329.83]
 [328.34]
 [322.04]]
```

```
#Visualizing the Data
plt.plot(training_set, color = 'red', label = 'Google Stock Price in Test set')
plt.xlabel('Time')
plt.ylabel('Google Stock Price')
plt.legend()
plt.show()
```


```
#feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(training_set)
print(training_set_scaled[0:5])
```

```
[[0.08581368]
 [0.09701243]
 [0.09433366]
 [0.09156187]
 [0.07984225]]
```

```
#preaparing the dataset for Training
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)
```

```
print('X_train shape = {}'.format(X_train.shape))
print('y_train shape = {}'.format(y_train.shape))
```

```
X_train shape = (1198, 60)
y_train shape = (1198,)
```

```
#reshaping the input data to fit in Keras RNN
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_train.shape
```

```
(1198, 60, 1)
```

Model Development

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

```
from keras.layers import LSTM
from keras.layers import Dropout


#LSTM Layers with Dropout regularization
regressor = Sequential()
regressor.add(LSTM(units= 50, return_sequences=True, input_shape = (X_train.shape
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50, return_sequences= True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50, return_sequences= True))
regressor.add(Dropout(0.2))
regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))

#Output Layer
regressor.add(Dense(units=1))

#Compiling the model
regressor.compile(optimizer='adam', loss='mean_squared_error')

#fitting the model
regressor.fit(X_train, y_train, epochs=100, batch_size=32)
```

```
38/38 [==============================] - 4s 113ms/step - loss: 0.0018
Epoch 72/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0018
Epoch 73/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0020
Epoch 74/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0018
Epoch 75/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0021
Epoch 76/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0018
Epoch 77/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0018
Epoch 78/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0019
Epoch 79/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0018
Epoch 80/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0020
Epoch 81/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0016
Epoch 82/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0019
Epoch 83/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0017
Epoch 84/100
38/38 [==============================] - 4s 115ms/step - loss: 0.0022
Epoch 85/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0018
```

```
Epoch 86/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0018
Epoch 87/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0019
Epoch 88/100
38/38 [==============================] - 4s 116ms/step - loss: 0.0016
Epoch 89/100
38/38 [==============================] - 5s 120ms/step - loss: 0.0018
Epoch 90/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0017
Epoch 91/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0015
Epoch 92/100
38/38 [==============================] - 4s 118ms/step - loss: 0.0015
Epoch 93/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0014
Epoch 94/100
38/38 [==============================] - 5s 119ms/step - loss: 0.0017
Epoch 95/100
38/38 [==============================] - 4s 117ms/step - loss: 0.0015
Epoch 96/100
38/38 [==============================] - 5s 119ms/step - loss: 0.0014
Epoch 97/100
38/38 [==============================] - 4s 118ms/step - loss: 0.0015
Epoch 98/100
38/38 [==============================] - 5s 119ms/step - loss: 0.0013
Epoch 99/100
38/38 [==============================] - 5s 118ms/step - loss: 0.0014
Epoch 100/100
```

```
#loading the Data
dataset_test = pd.read_csv('Google_Stock_Price_Test.csv')
real_stock_price = dataset_test.iloc[:,1:2].values


#preprocessing the Data
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))
```

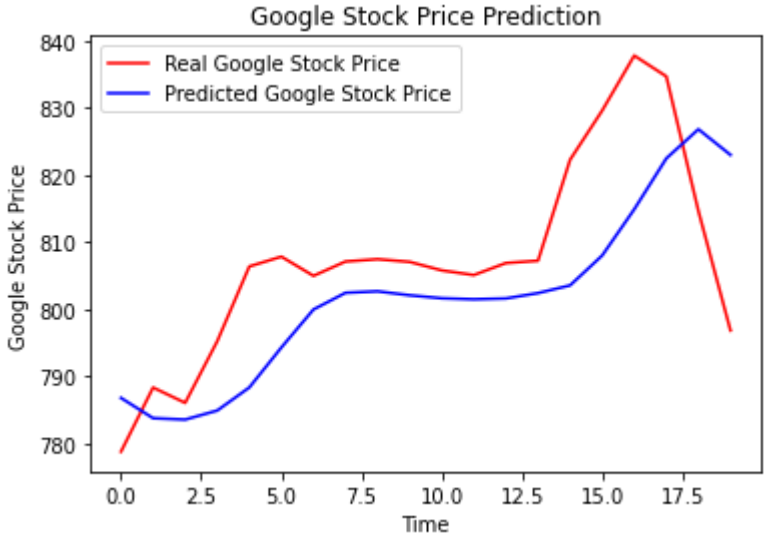
Output Prediction

```
#predicting the output
predicted_stock_price = regressor.predict(X_test)
predicted_stock_price = sc.inverse_transform(predicted_stock_price)
```

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

Result Visualization

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



✓ 0s completed at 18:12