Google Stock Prediction with RNN [ghoshs4@lsbu.ac.uk]

September 3, 2019

1 Part 1 : Data Pre-Processing

1.1 import basic Libraries

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

1.2 import training set

[2]:
```
actual_training_set = pd.read_csv('Google_Stock_Price_Train.csv')
```

Sample data set.

[3]:
```
actual_training_set.head()
```

[3]:
```
       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
```

Data dimension

[4]:
```
actual_training_set.shape
```

[4]:
```
(1258, 6)
```

1.3 Extract Training Set

each attributes of the dataset is a feature and time series. for the experiement we're extracting the **Open** column and updating the training set with it and save it in a **matrix format**.

[5]:
```
#this process is complicated for our purpose
#training_set=actual_training_set[['Open']]

training_set=actual_training_set.iloc[:,1:2].values  #:2 making it a matrix
```

[6]:
```
training_set
```

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

check the dimension.

```
[7]: training_set.shape
```

```
[7]: (1258, 1)
```

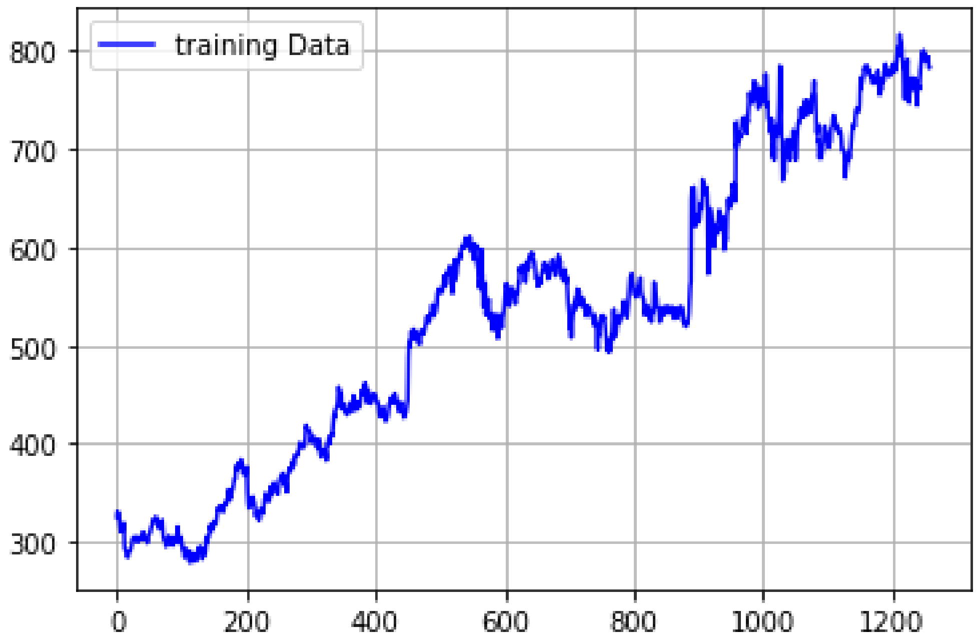
The plot of the current Dataset

```
[8]: def plot_me(vector,style=[],lab=[]):
         %matplotlib inline
         plt.grid(True)

         if len(vector) == len(style) == len(lab):
             for i in range(len(vector)):
                 plt.plot(np.arange(len(vector[i])), vector[i],
                          style[i], label=lab[i])
         else:
             print('Error : dimention error! ')

         plt.legend()
         plt.show()
```

```
[9]: plot_me(vector=[training_set],
             style=['b-'],
             lab=['training Data'])
```



1.4 Feature Scalling

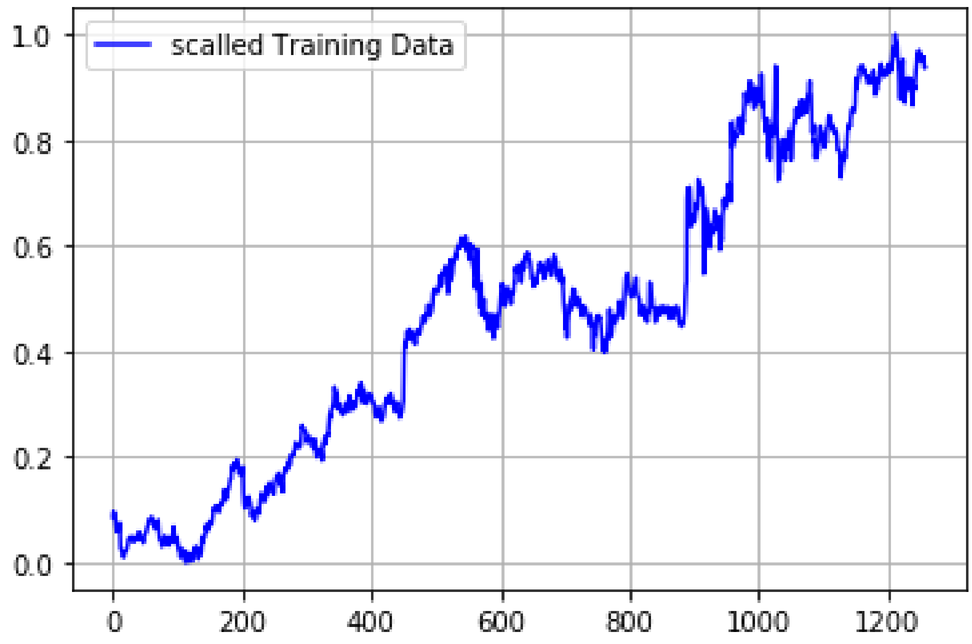
there are two options for feature scalling 1. Standardisaton : $X_{stand} = \frac{X-\mu_X}{\sigma_X}$ 2. Normalization : $X_{norm} = \frac{X-Min(X)}{Max(X)-Min(X)}$

we'll apply both

[10]:
```
#Apply Normalization
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler()
training_set_scalled=sc.fit_transform(training_set)
```

plot of scalled Training data

[11]:
```
plot_me(vector=[training_set_scalled],
        style=['b-'],
        lab=['scalled Training Data'])
```



1.5 Train-Test Split

if you have stock price in time t, you're predicting stock price at time $t + 1$.

[12]:
```
X_train = training_set_scalled[:-1] #all stock price except last
y_train = training_set_scalled[1:] #stock price shifted by 1
```

1.6 Reshapping

the purpose of reshapping is to change the dimention from 2D to 3D, the 3rd dimention is needed to make it a **Tensor** as to be compatible with Keras moreover Tensorflow input format. `(batch_size, time_step, feature)`

[13]:
```
X_train_t = np.reshape(X_train, (X_train.shape[0], #obs
                                 X_train.shape[1], #ts
                                 1)                #feature
                      )
```

[14]:
```
X_train_t.shape
```

[14]:
```
(1257, 1, 1)
```

[15]:
```
y_train.shape
```

[15]:
```
(1257, 1)
```

[16]:
```
X_train
```

[16]:
```
array([[0.08581368],
       [0.09701243],
       [0.09433366],
       ...,
       [0.95163331],
       [0.95725128],
       [0.93796041]])
```

[17]:
```
y_train
```

[17]:
```
array([[0.09701243],
       [0.09433366],
       [0.09156187],
       ...,
       [0.95725128],
       [0.93796041],
       [0.93688146]])
```

2 Part 2 : Building the RNN

2.1 import keras libraries

[18]:
```
from keras.models import Sequential
from keras.layers import LSTM
from keras.layers import Dense
from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

2.2 Initialise Deep Regressor

[19]:
```
#init RNN
regressor = Sequential()
```

2.3 Adding Layers

[20]:
```
#Adding input layers
regressor.add(LSTM(units = 4, #number of memory units
              activation = 'sigmoid',  #sigmoid
              input_shape = (None, 1)) #(time_step, feature)
             )

#Adding output layer
regressor.add(Dense(units = 1)) #since predicting one value
```

```
WARNING:tensorflow:From C:\Users\sapta\Anaconda3\lib\site-
packages\tensorflow\python\framework\op_def_library.py:263: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
```

```
version.
Instructions for updating:
Colocations handled automatically by placer.
```

2.4 Compile The Model

[21]:
```
#Compile the model
regressor.compile(optimizer = 'adam', #rmsprop gives same out but more memory
                  loss='mean_squared_error')
```

2.5 Train the Model

[22]:
```
import time
```

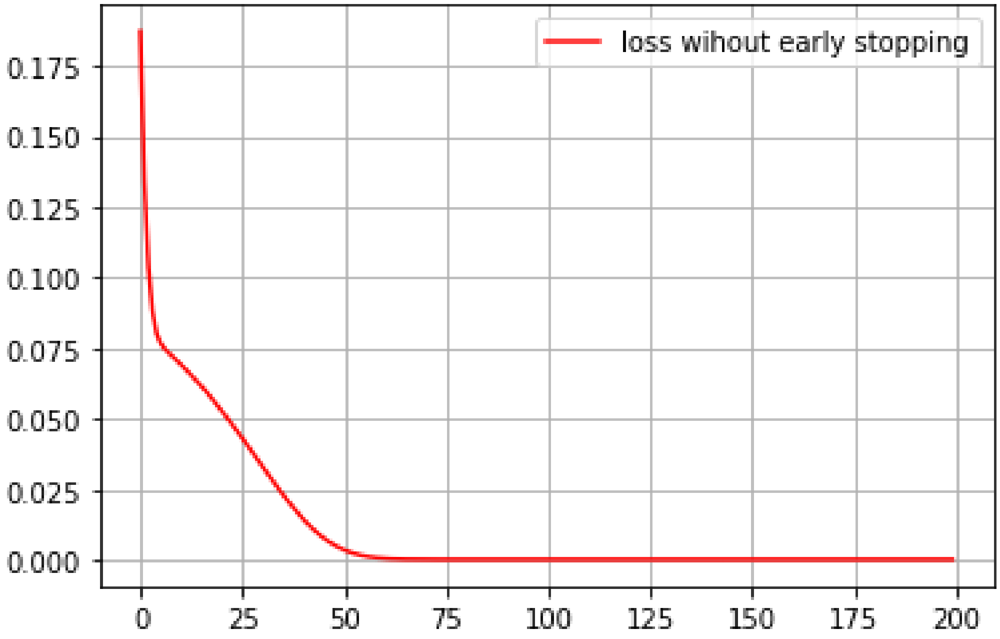
[23]:
```
#Train the network
t =time.time()

learn_his = regressor.fit(X_train_t, y_train, batch_size=32, epochs=200,␣
 ↪verbose=False)

print(f'time taken without early stopping : {round(time.time()-t,3)} secs')

#plot Loss
plt.grid()
plt.plot(learn_his.history['loss'],'r-',label = 'loss wihout early stopping')
plt.legend()
plt.show()
```

```
WARNING:tensorflow:From C:\Users\sapta\Anaconda3\lib\site-
packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.cast instead.
time taken without early stopping : 26.552 secs
```



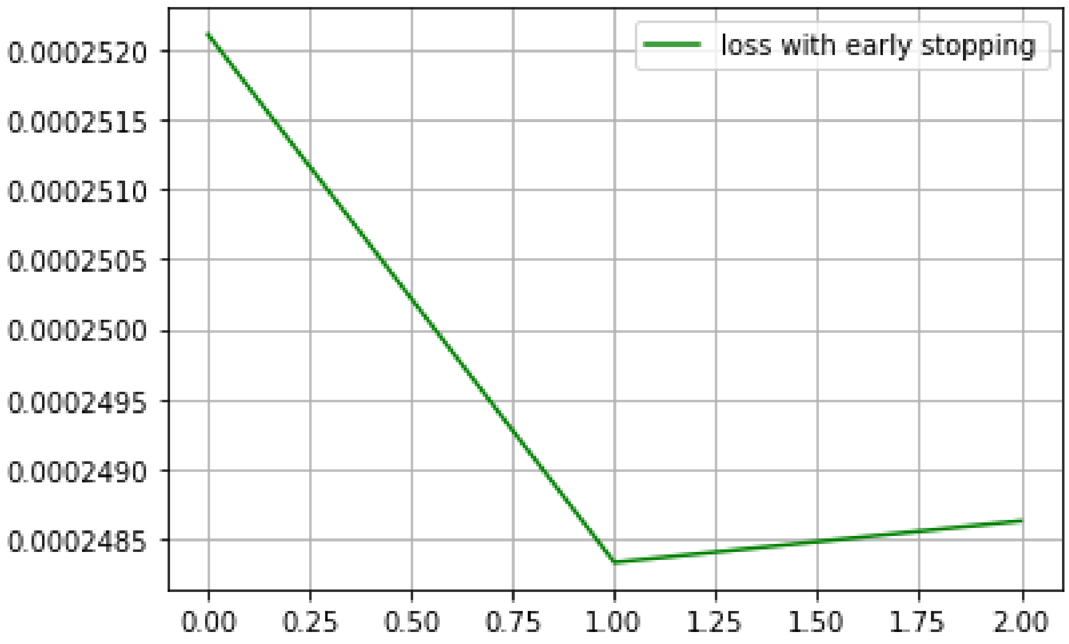
2.5.1 Speeding up learning process

[33]:

```
#Train the network
t =time.time()
early_stop = EarlyStopping(monitor='loss', patience=1, verbose=0)
learn_his = regressor.fit(X_train_t, y_train, batch_size=32, epochs=200,␣
 ↪verbose=False, callbacks=[early_stop])
print(f'time taken without early stopping : {round(time.time()-t,3)} Secs')

#plot Loss
plt.grid()
plt.plot(learn_his.history['loss'],'g-',label = 'loss with early stopping')
plt.legend()
plt.show()
```

```
time taken without early stopping : 0.338 Secs
```



3 Part 3 : make Forecast

3.1 Build a test Set

[25]:

```
#Test Set
test_set = pd.read_csv('Google_stock_Price_Test.csv')
X_test = test_set.iloc[:,1:2].values
```

3.2 Make Prediction

[26]:

```
# Make prediction
X_test_scalled = sc.transform(X_test)
X_test_t = np.reshape(X_test_scalled, (X_test.shape[0],
                                       X_test.shape[1],
                                       1)
                      )
```

3.2.1 Pre-Scalling

[27]:

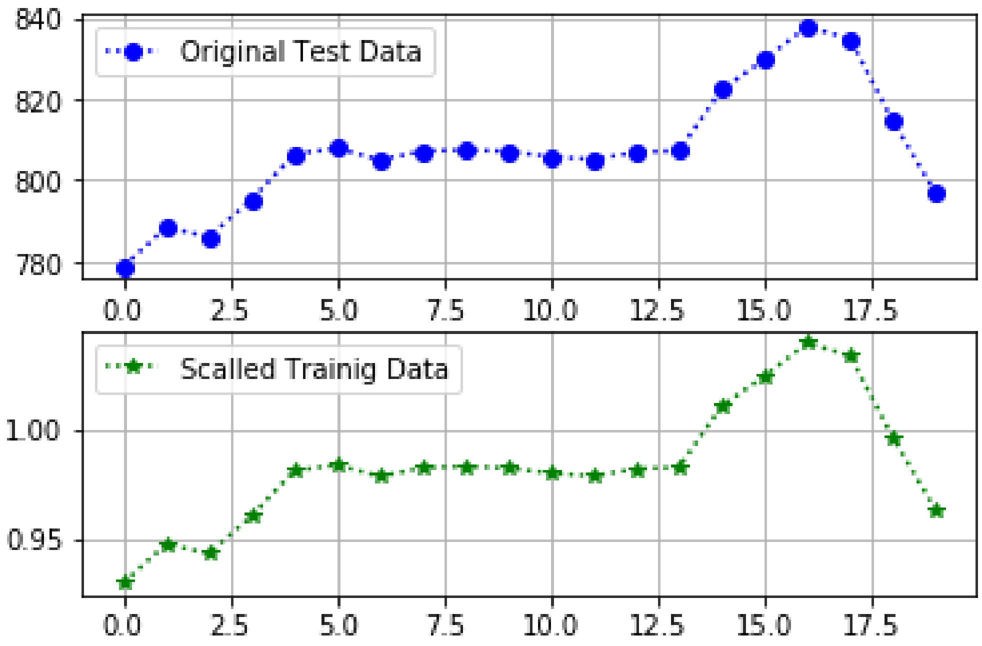
```
ax1 = plt.subplot('211')
ax2 = plt.subplot('212')

ax1.grid()
ax2.grid()
```

```
ax1.plot(X_test,'b:o' , label = 'Original Test Data')
ax2.plot(X_test_scalled,'g:*' , label = 'Scalled Trainig Data')

ax1.legend()
ax2.legend()

plt.show()
```



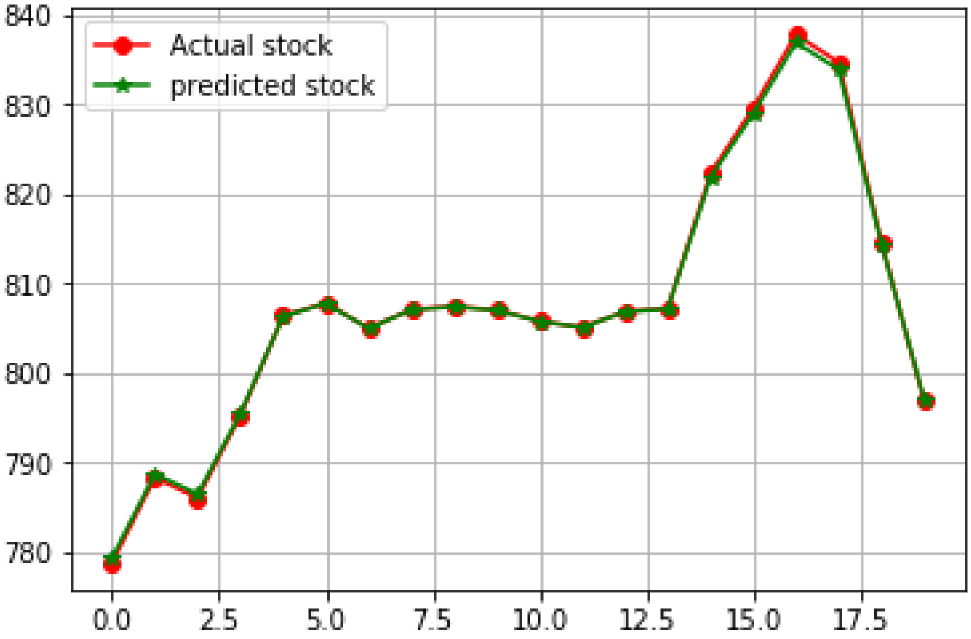
[28]:

```
y_pred = regressor.predict(X_test_t)
y_pred_org = sc.inverse_transform(y_pred) # restore original scale
```

3.3 Plot the graph

[36]:

```
#Plot
plot_me(vector = [X_test,y_pred_org],
        style = ['r-o','g-*'],
        lab = ['Actual stock','predicted stock'])
```



[31]:

```
from platform import python_version
print(python_version())
```

```
3.7.3
```