Data Science For Everyone Using Python

Manaranjan Pradhan
manaranjan@enablecloud.com
*This notebook is given as part of **Data Science for everyone** workshop.*

Text Analytics - Sentiment Analysis

In [2]:

```
import pandas as pd
import numpy as np
```

In [3]:

```
train_ds = pd.read_csv( "sentiment_train", delimiter="\t" )
```

In [4]:

```
train_ds.head( 10 )
```

Out[4]:

	sentiment	text
0	1	The Da Vinci Code book is just awesome.
1	1	this was the first clive cussler i've ever rea...
2	1	i liked the Da Vinci Code a lot.
3	1	i liked the Da Vinci Code a lot.
4	1	I liked the Da Vinci Code but it ultimatly did...
5	1	that's not even an exaggeration) and at midni...
6	1	I loved the Da Vinci Code, but now I want some...
7	1	i thought da vinci code was great, same with k...
8	1	The Da Vinci Code is actually a good movie...
9	1	I thought the Da Vinci Code was a pretty good ...

In [5]:

```
train_ds.shape
```

Out[5]:

```
(6918, 2)
```

In [7]:

```
from sklearn.feature_extraction.text import CountVectorizer
```

In [8]:

```
count_vectorizer = CountVectorizer( max_features = 5000 )
```

In [9]:

```
feature_vector = count_vectorizer.fit( train_ds.text )
train_ds_features = count_vectorizer.transform( train_ds.text )
```

In [10]:

```
features = feature_vector.get_feature_names()
```

In [11]:

```
features_counts = np.sum( train_ds_features.toarray(), axis = 0 )
```

In [12]:

```
feature_counts = pd.DataFrame( dict( features = features,
                                     counts = features_counts ) )
```

In [13]:

```
feature_counts.head(5)
```

Out[13]:

	counts	features
0	1	00
1	1	007
2	4	10
3	1	10pm
4	1	12

In [17]:

```
feature_counts.sort( "counts", ascending = False )[1:20]
```

Out[17]:

	counts	features
93	2154	and
864	2093	harry
1466	2093	potter
355	2002	code
2009	2001	vinci
442	2001	da
1272	2000	mountain
259	2000	brokeback
1171	1624	love
1018	1520	is
2029	1176	was
151	1127	awesome
1252	1094	mission
977	1093	impossible
1132	974	like
1022	901	it
1916	808	to
1275	783	movie
1862	719	that

In [18]:

```
count_vectorizer = CountVectorizer( stop_words = "english",
                                    max_features = 5000 )
feature_vector = count_vectorizer.fit( train_ds.text )
train_ds_features = count_vectorizer.transform( train_ds.text )
```

In [20]:

```
features = feature_vector.get_feature_names()
features_counts = np.sum( train_ds_features.toarray(), axis = 0 )
feature_counts = pd.DataFrame( dict( features = features,
                                     counts = features_counts ) )
feature_counts.sort( "counts", ascending = False )[0:20]
```

Out[20]:

	counts	features
1328	2093	potter
790	2093	harry
314	2002	code
1823	2001	vinci
399	2001	da
1167	2000	mountain
223	2000	brokeback
1074	1624	love
126	1127	awesome
1150	1094	mission
892	1093	impossible
1035	974	like
1169	783	movie
1646	602	sucks
1644	600	sucked
792	578	hate
1393	374	really
1170	366	movies
1637	365	stupid
967	287	just

In [21]:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.cross_validation import train_test_split
```

In [22]:

```
clf = GaussianNB()
```

In [23]:

```
train_X, test_X, train_y, test_y = train_test_split( train_ds_features,
                                                     train_ds.sentiment,
                                                     test_size = 0.3,
                                                     random_state = 42 )
```

In [24]:

```
clf.fit( train_X.toarray(), train_y )
```

Out[24]:

```
GaussianNB()
```

In [25]:

```
test_ds_predicted = clf.predict( test_X.toarray() )
```

In [26]:

```
from sklearn import metrics
```

In [27]:

```
cm = metrics.confusion_matrix( test_y, test_ds_predicted )
```

In [28]:

```
cm
```

Out[28]:

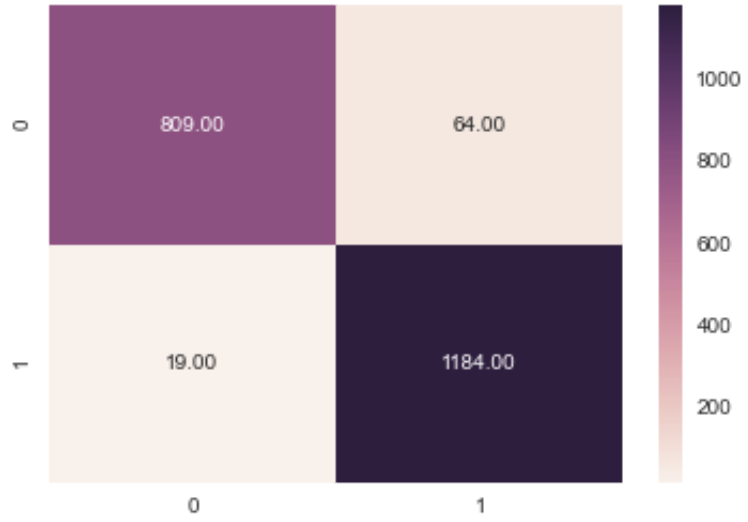
```
array([[ 809,   64],
       [  19, 1184]])
```

In [38]:

```
import matplotlib as plt
import seaborn as sn
%matplotlib inline
```

In [39]:

```
sn.heatmap(cm, annot=True,  fmt='.2f' );
```



In [40]:

```
score = metrics.accuracy_score( test_y, test_ds_predicted )
```

In [41]:

```
score
```

Out[41]:

0.96001926782273606

In [42]:

```
# read the entire file into a python array
with open('azhar.json', 'r') as f:
    data = f.readlines()

# remove the trailing "\n" from each line
data = map(lambda x: x.rstrip(), data)
```

In [43]:

```
data_json_str = "[" + ','.join(data) + "]"
```

In [47]:

```
azhar_df = pd.read_json(data_json_str)
```

In [46]:

```
azhar_df.head( 2 )
```

Out[46]:

	contributors	coordinates	created_at	entities	extended_entities
0	NaN	NaN	2016-05-13 08:26:27	{'user_mentions': [{'screen_name': 'bookmyshow...	NaN
1	NaN	NaN	2016-05-13 08:26:28	{'user_mentions': [{'screen_name': 'bookmyshow...	NaN

2 rows × 31 columns

In [48]:

```
azhar_df = azhar_df[['text']]
```

In [49]:

```
azhar_df.head( 2 )
```

Out[49]:

	text
0	RT @bookmyshow: 8. Name the city #Azhar is fro...
1	RT @bookmyshow: 3. True Or False: @ItsPrachiDe...

In [50]:

```
azhar_df = azhar_df[-azhar_df.text.str.contains( "@bookmyshow" )]
```

In [51]:

```
azhar_df.head( 2 )
```

Out[51]:

	text
6	RT @bollywood_life: @emraanhashmi hits a sixer...
9	RT @taran_adaarsh: #AZHAR is Outstanding..Don'...

In [52]:

```
azhar_text = count_vectorizer.transform( azhar_df.text )
```

In [53]:

```
azhar_text[1]
```

Out[53]:

```
<1x1921 sparse matrix of type '<class 'numpy.int64'>'
        with 8 stored elements in Compressed Sparse Row format>
```

In [54]:

```
azhar_df["sentiment"] = clf.predict( azhar_text.toarray() )
```

In [57]:

```
azhar_df[0:10]
```

Out[57]:

	text	sentiment
6	RT @bollywood_life: @emraanhashmi hits a sixer...	0
9	RT @taran_adaarsh: #AZHAR is Outstanding..Don'...	1
10	RT @ursmehreen: Omg! Today is Friday! #Azhar r...	0
11	RT @taran_adaarsh: #AZHAR is Outstanding..Don'...	1
13	RT @girishjohar: #Azhar starts on a comfortabl...	1
14	RT @bobbytalkcinema: AZHAR - Interesting twist...	0
16	Azhar Movie Review and Rating Hit or Flop Publ...	0
17	RT @itimestweets: Live #Azhar review: @emraanh...	0
19	RT @rajcheerfull: Looking forward to #Azhar	1
20	@TrollKejri your review on #Azhar #Azharthefilm	0

In [56]:

```
azhar_df.to_csv( "azhar_sentiments.csv", index = False )
```