Text_Analytics_Sentiment_SVG

August 21, 2020

[283]:
```
#Text Analytics - Classification using Naive Bayes Technique for the Data from:
#Amazon - eCommerce company,
#IMDB - is the authoritative source for movie, TV and celebrity content and␣
 ↪ratings and reviews,
#Yelp - On line Advertising Company's
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
from sklearn.feature_extraction.text import CountVectorizer
import random
from sklearn.metrics import roc_curve
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction import text
from nltk.stem.snowball import PorterStemmer
from sklearn.naive_bayes import BernoulliNB
from sklearn import metrics
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import GaussianNB
from nltk.stem import PorterStemmer
from nltk import word_tokenize
import re
import warnings
warnings.filterwarnings('ignore')
```

[211]:
```
#importing Amazon data
sentiment1_df = pd.read_csv("C:/Venu/UCI DataSets/sentiment labelled sentences/
 ↪amazon_cells_labelled.txt", sep = "\t",header=None,names =␣
 ↪("Text","Sentiment"))
```

[212]:
```
sentiment1_df.head(10)
```

[212]:

	Text	Sentiment
0	So there is no way for me to plug it in here i…	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1

```
3  Tied to charger for conversations lasting more…          0
4                                The mic is great.          1
5  I have to jiggle the plug to get it to line up…          0
6  If you have several dozen or several hundred c…          0
7        If you are Razr owner…you must have this!          1
8                Needless to say, I wasted my money.          0
9                   What a waste of money and time!.          0
```

[213]:
```
sentiment1_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Text       1000 non-null   object
 1   Sentiment  1000 non-null   int64
dtypes: int64(1), object(1)
memory usage: 15.8+ KB
```

[214]:
```
#Reading IMDB data
sentiment2_df=pd.read_csv("C:/Venu/UCI DataSets/sentiment labelled sentences/
 ↪imdb_labelled.txt",sep='\t',header=None,names=("Text","Sentiment"))
```

[215]:
```
sentiment2_df.head(10)
```

[215]:
```
                                                Text  Sentiment
0  A very, very, very slow-moving, aimless movie …          0
1  Not sure who was more lost - the flat characte…          0
2  Attempting artiness with black & white and cle…          0
3       Very little music or anything to speak of.          0
4  The best scene in the movie was when Gerardo i…          1
5  The rest of the movie lacks art, charm, meanin…          0
6                            Wasted two hours.          0
7  Saw the movie today and thought it was a good …          1
8                           A bit predictable.          0
9  Loved the casting of Jimmy Buffet as the scien…          1
```

[216]:
```
sentiment2_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 748 entries, 0 to 747
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Text       748 non-null    object
 1   Sentiment  748 non-null    int64
```

```
dtypes: int64(1), object(1)
memory usage: 11.8+ KB
```

[217]:
```
#Reading Yelp Data
sentiment3_df=pd.read_csv("C:/Venu/UCI DataSets/sentiment labelled sentences/
 ↪yelp_labelled.txt",sep='\t',header=None,names=("Text","Sentiment"))
```

[218]:
```
sentiment3_df.head(10)
```

[218]:

	Text	Sentiment
0	Wow… Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of…	1
4	The selection on the menu was great and so wer…	1
5	Now I am getting angry and I want my damn pho.	0
6	Honeslty it didn't taste THAT fresh.)	0
7	The potatoes were like rubber and you could te…	0
8	The fries were great too.	1
9	A great touch.	1

[219]:
```
sentiment3_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Text       1000 non-null   object
 1   Sentiment  1000 non-null   int64
dtypes: int64(1), object(1)
memory usage: 15.8+ KB
```

[220]:
```
#Appending Amazon and IMDB data
sentiment_df=sentiment1_df.append(sentiment2_df,ignore_index=True)
```

[221]:
```
sentiment_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1748 entries, 0 to 1747
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Text       1748 non-null   object
 1   Sentiment  1748 non-null   int64
dtypes: int64(1), object(1)
memory usage: 27.4+ KB
```

[222]:
```
#Appending Yelp Data and preparing final dataset for analysis
sentiment_df=sentiment_df.append(sentiment3_df,ignore_index=True)
```

[223]:
```
#Exploring Text Data Set
sentiment_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2748 entries, 0 to 2747
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Text       2748 non-null   object
 1   Sentiment  2748 non-null   int64
dtypes: int64(1), object(1)
memory usage: 43.1+ KB
```

[224]:
```
sentiment_df.Sentiment.count()
```

[224]:
```
2748
```

[225]:
```
sentiment_df[sentiment_df.Sentiment==1]
```

[225]:

	Text	Sentiment
1	Good case, Excellent value.	1
2	Great for the jawbone.	1
4	The mic is great.	1
7	If you are Razr owner…you must have this!	1
10	And the sound quality is great.	1
…	…	…
2647	Overall, a great experience.	1
2649	Their regular toasted bread was equally satisf…	1
2655	The chips and sals a here is amazing!!!!!!!!!!…	1
2657	This is my new fav Vegas buffet spot.	1
2670	Every time I eat here, I see caring teamwork t…	1

[1386 rows x 2 columns]

[226]:
```
sentiment_df[sentiment_df.Sentiment==0]
```

[226]:

	Text	Sentiment
0	So there is no way for me to plug it in here i…	0
3	Tied to charger for conversations lasting more…	0
5	I have to jiggle the plug to get it to line up…	0
6	If you have several dozen or several hundred c…	0
8	Needless to say, I wasted my money.	0
…	…	…
2743	I think food should have flavor and texture an…	0

```
2744                            Appetite instantly gone.          0
2745  Overall I was not impressed and would not go b…          0
2746  The whole experience was underwhelming, and I …          0
2747  Then, as if I hadn't wasted enough of my life …          0

[1362 rows x 2 columns]
```

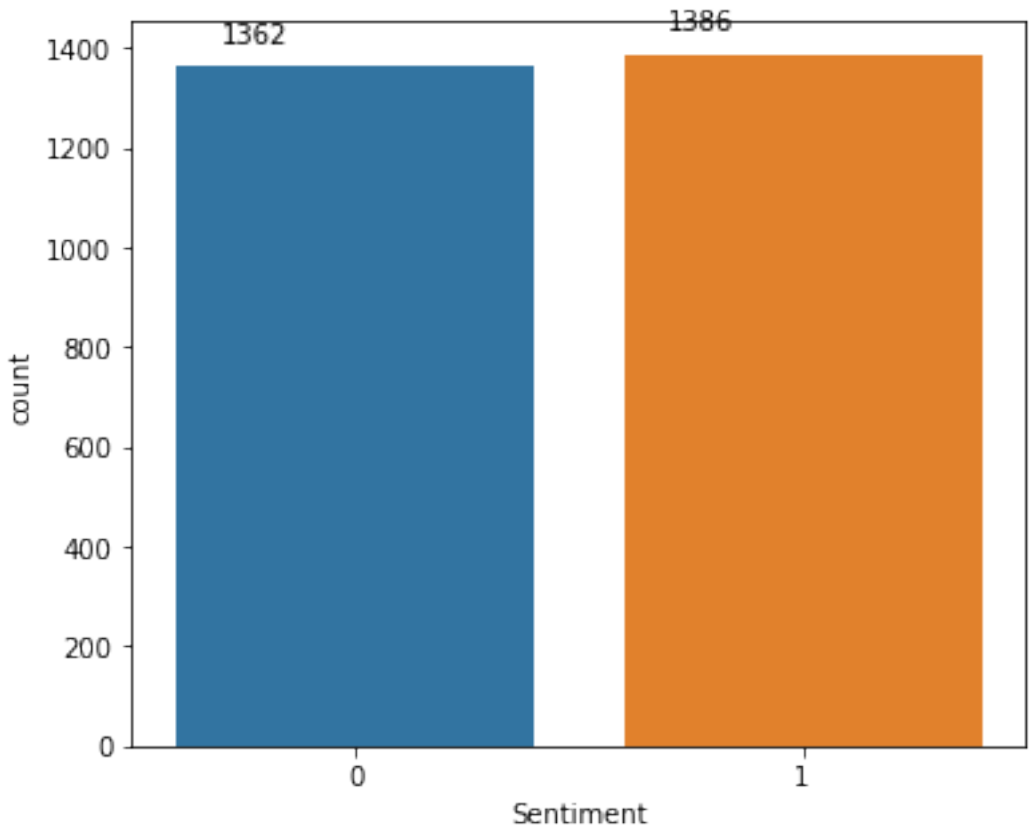
[227]:
```
sentiment_df[sentiment_df.Sentiment==1][0:5]
```

[227]:
```
                                          Text  Sentiment
1                    Good case, Excellent value.          1
2                         Great for the jawbone.          1
4                              The mic is great.          1
7   If you are Razr owner…you must have this!          1
10               And the sound quality is great.          1
```

[228]:
```
sentiment_df[sentiment_df.Sentiment==0][0:5]
```

[228]:
```
                                                Text  Sentiment
0  So there is no way for me to plug it in here i…          0
3  Tied to charger for conversations lasting more…          0
5  I have to jiggle the plug to get it to line up…          0
6  If you have several dozen or several hundred c…          0
8                Needless to say, I wasted my money.          0
```

[229]:
```
#Ploting Sentiment counts for the final dataset
plt.figure(figsize=(6,5))
ax = sn.countplot(x='Sentiment',data=sentiment_df)
for p in ax.patches:
    ax.annotate(p.get_height(),(p.get_x()+0.1,p.get_height()+50))
```



[230]:
```
##Count Vector Model
# Initialize the Count Vectorizer
count_vectorizer = CountVectorizer()
# Create the dictionary from the dataset
feature_vector = count_vectorizer.fit(sentiment_df.Text)
#Get the feature names
features = feature_vector.get_feature_names()
print("Total number of features:", len(features))
```

```
Total number of features: 5155
```

[231]:
```
#Listing some of the features in the Data set
random.sample(features,10)
```

[231]:
```
['remorse',
 'spock',
 'dying',
 'boot',
 'personally',
 'howe',
```

```
 'think',
 'guys',
 'gung',
 'my']
```

[232]:
```
#Creating a matrix (sparse) of  documents x features
sentiment_df_features = count_vector.transform(sentiment_df.Text)
type(sentiment_df_features)
```

[232]:
```
scipy.sparse.csr.csr_matrix
```

[233]:
```
#Dimensions of the transformed matrix of documents x features
sentiment_df_features.shape
```

[233]:
```
(2748, 5155)
```

[234]:
```
#Get non zeros values from the Sparse matrix
sentiment_df_features.getnnz()
```

[234]:
```
30275
```

[235]:
```
#Density of the matrix
print("Density = ",sentiment_df_features.getnnz()*100/(sentiment_df_features.
 ↪shape[0]*sentiment_df_features.shape[1]))
```

```
Density =  0.21371684476991995
```

[236]:
```
#displaying document vectors
#Converting the matrix to a dataframe
sentiment_features_df = pd.DataFrame(sentiment_df_features.todense())
```

[237]:
```
sentiment_features_df.columns = features
```

[238]:
```
sentiment_features_df.columns
```

[238]:
```
Index(['00', '10', '100', '11', '12', '13', '15', '15g', '15pm', '17',
       …
       'yucky', 'yukon', 'yum', 'yummy', 'yun', 'z500a', 'zero', 'zillion',
       'zombie', 'zombiez'],
      dtype='object', length=5155)
```

[240]:
```
sentiment_df[0:1]
```

[240]:

	Text	Sentiment
0	So there is no way for me to plug it in here i…	0

[241]:
```
sentiment_features_df.iloc[0:1, 2150:2200]
```

[241]:
```
   help  helped  helpful  helping  helps  hence  hendrikson  her  here  \
0     0       0        0        0      0      0           0    0     1

   hereas  …  hits  ho  hockey  hoffman  hold  holder  holding  holds  hole  \
0       0  …     0   0       0        0     0       0        0      0     0

   holes
0      0

[1 rows x 50 columns]
```

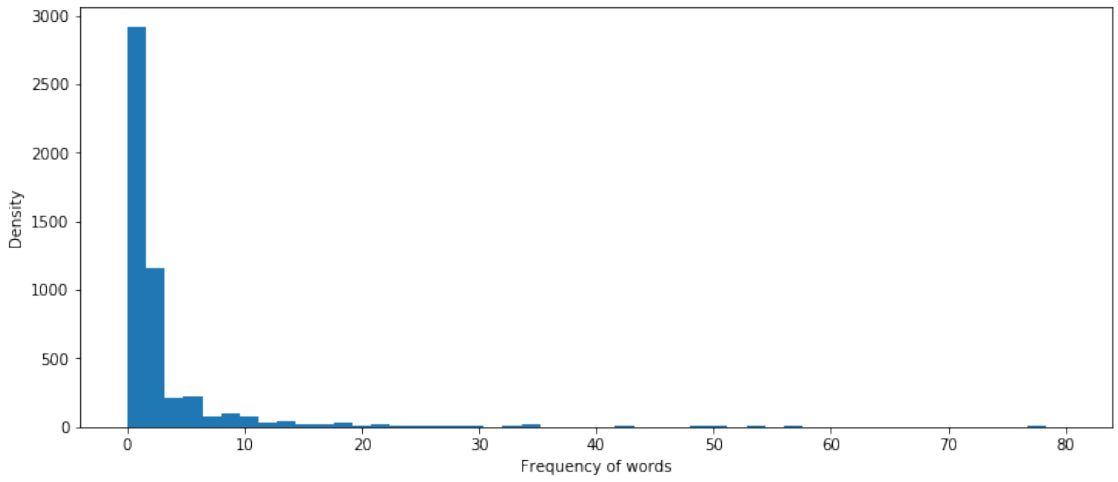
[242]:
```
sentiment_features_df[['so','there','is','no','way','for','me','to','plug','it','in','here']][
↪1]
```

[242]:
```
   so  there  is  no  way  for  me  to  plug  it  in  here
0   1      1   1   1    1    1   1   1     1   1   2     1
```

[243]:
```
#Removing Low-Frequency words
features_counts = np.sum(sentiment_df_features.toarray(),axis=0)
feature_counts_df = pd.DataFrame(dict(features =␣
↪features,counts=features_counts))
```

[244]:
```
plt.figure(figsize=(12,5))
plt.hist(feature_counts_df.counts,bins=50,range=(0,80))
plt.xlabel('Frequency of words')
plt.ylabel('Density')
```

[244]:
```
Text(0, 0.5, 'Density')
```



[245]:
```
len(feature_counts_df[feature_counts_df.counts==1])
```

[245]: 2918

[246]:
```
#set the max_features =2500; as 2918 features occurs only once out of 5155
count_vector1 = CountVectorizer(max_features=2500)
feature_vector1 = count_vector1.fit(sentiment_df.Text)
features1 = feature_vector1.get_feature_names()
sentiment_df_features1 = count_vector1.transform(sentiment_df.Text)
features1_counts = np.sum(sentiment_df_features1.toarray(),axis=0)
features1_counts = pd.DataFrame(dict(features1 = features1, counts =␣
↪features1_counts))
features1_counts.sort_values('counts',ascending = False)[0:15]
```

[246]:

	features1	counts
2197	the	1953
85	and	1138
1036	it	789
1033	is	754
2238	to	670
2215	this	643
1334	of	624
2395	was	571
1005	in	400
801	for	336
2195	that	316
1315	not	306
2455	with	274
1281	my	254
2365	very	245

[247]:
```
#Removing Stop words such as if, not, ...
my_stop_words = text.ENGLISH_STOP_WORDS
print('Few stop words: ', list(my_stop_words)[0:10])
```

```
Few stop words:  ['besides', 'describe', 'from', 'together', 'and', 'such',
'due', 'former', 'keep', 'neither']
```

[248]:
```
#set the max_features =2500; as 2918 features occurs only once out of 5155 and␣
↪removing stop words list
count_vector2 = CountVectorizer(stop_words=my_stop_words, max_features=2500)
feature_vector2 = count_vector2.fit(sentiment_df.Text)
features2 = feature_vector2.get_feature_names()
sentiment_df_features2 = count_vector2.transform(sentiment_df.Text)
features2_counts = np.sum(sentiment_df_features2.toarray(),axis=0)
features2_counts = pd.DataFrame(dict(features2 = features2, counts =␣
↪features2_counts))
features2_counts.sort_values('counts',ascending = False)[0:15]
```

[248]:

	features2	counts
786	good	230
793	great	210
1152	movie	182
1328	phone	168
689	film	163
721	food	126
1014	like	125
957	just	119
1357	place	114
2039	time	112
1788	service	108
127	bad	103
1621	really	103
530	don	80
163	best	78

[249]:
```
#Stemming and Lemmatization
stemmer = PorterStemmer()
analyzer = CountVectorizer().build_analyzer()
def stemmed_words(doc):
    #Stemming of words
    stemmed_words = [stemmer.stem(w) for w in analyzer(doc)]
    ### Removing the words in stop words list
    non_stop_words = [word for word in list(set(stemmed_words) -␣
 ↪set(my_stop_words))]
    return non_stop_words
```

[250]:
```
#set the max_features =2500; as 2918 features occurs only once out of 5155,␣
 ↪removing stop words list and stemmed words
count_vector3 = CountVectorizer(analyzer = stemmed_words, max_features=2500)
feature_vector3 = count_vector3.fit(sentiment_df.Text)
features3 = feature_vector3.get_feature_names()
sentiment_df_features3 = count_vector3.transform(sentiment_df.Text)
features3_counts = np.sum(sentiment_df_features3.toarray(),axis=0)
features3_counts = pd.DataFrame(dict(features3 = features3, counts =␣
 ↪features3_counts))
features3_counts.sort_values('counts',ascending = False)[0:15]
```

[250]:

	features3	counts
2039	thi	552
2346	wa	472
2302	veri	218
695	good	210
702	great	191
1133	phone	167

```
1018      movi     147
615       film     140
2451      work     136
2069      time     129
900       like     126
1153     place     122
639       food     122
1618    servic     109
854       just     105
```

[251]:
```
sentiment4_df = pd.DataFrame(sentiment_df_features.todense())
sentiment4_df.columns = features
sentiment4_df['Sentiment'] = sentiment_df.Sentiment
```

[252]:
```
sentiment4_df[sentiment4_df.Sentiment==0]
```

[252]:
```
      00  10  100  11  12  13  15  15g  15pm  17  …  yukon  yum  yummy  yun  \
0      0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
3      0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
5      0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
6      0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
8      0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
…    ..  ..  …  ..  ..  ..  ..  …    …  ..  …    …  …    …  …
2743   0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
2744   0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
2745   0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
2746   0   0    0   0   0   0   0    0     0   0  …      0    0      0    0
2747   0   0    0   0   0   0   0    0     0   0  …      0    0      0    0

      z500a  zero  zillion  zombie  zombiez  Sentiment
0         0     0        0       0        0          0
3         0     0        0       0        0          0
5         0     0        0       0        0          0
6         0     0        0       0        0          0
8         0     0        0       0        0          0
…     …   …       …      …       …        …
2743      0     0        0       0        0          0
2744      0     0        0       0        0          0
2745      0     0        0       0        0          0
2746      0     0        0       0        0          0
2747      0     0        0       0        0          0

[1362 rows x 5156 columns]
```

[253]:
```
sentiment4_df[sentiment4_df.Sentiment==1]
```

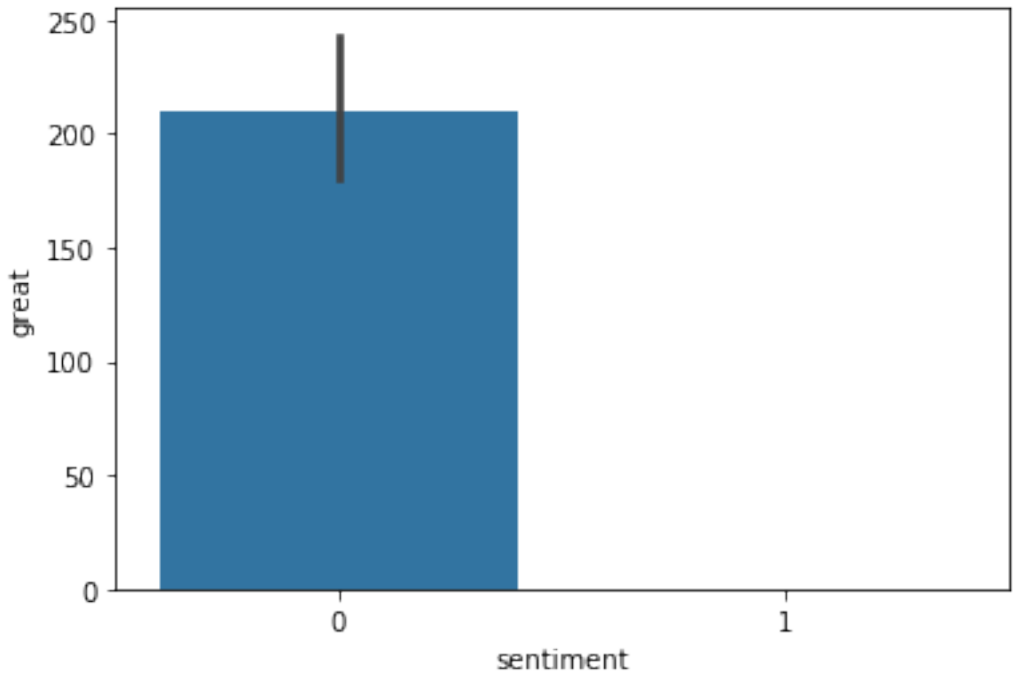
[253]:

	00	10	100	11	12	13	15	15g	15pm	17	…	yukon	yum	yummy	yun	\
1	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
…	..	..	…	..	..	..	..	…	…	..	…	…	…	…	…	
2647	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
2649	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
2655	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
2657	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	
2670	0	0	0	0	0	0	0	0	0	0	…	0	0	0	0	

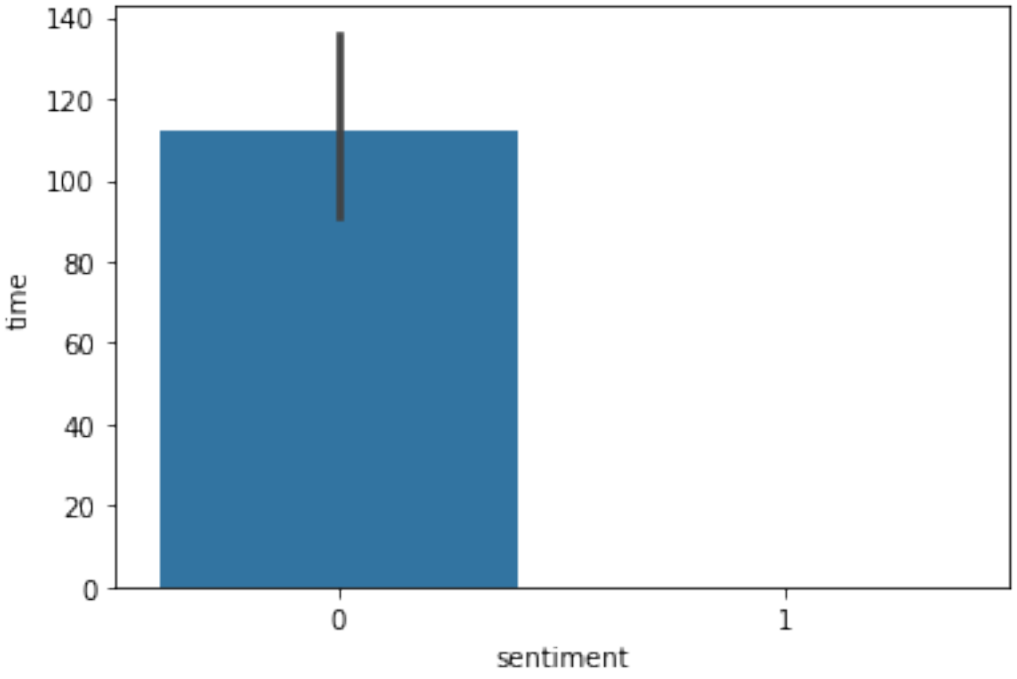
	z500a	zero	zillion	zombie	zombiez	Sentiment
1	0	0	0	0	0	1
2	0	0	0	0	0	1
4	0	0	0	0	0	1
7	0	0	0	0	0	1
10	0	0	0	0	0	1
…	…	…	…	…	…	
2647	0	0	0	0	0	1
2649	0	0	0	0	0	1
2655	0	0	0	0	0	1
2657	0	0	0	0	0	1
2670	0	0	0	0	0	1

[1386 rows x 5156 columns]

[254]:
```
sn.barplot(x = 'sentiment',y='great',data=sentiment4_df,estimator=sum);
```



[255]:
```
sn.barplot(x = 'sentiment',y='time',data=sentiment4_df,estimator=sum);
```



[256]:
```
train_X,test_X,train_y,test_y = train_test_split(sentiment4_df,sentiment4_df.
↪Sentiment,test_size=0.3,random_state=42)
```

[257]:
```
test_y.count()
```

[257]: 825

[258]:
```
nb_clf = BernoulliNB()
nb_clf.fit(train_X,train_y)
```

[258]: BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)

[259]:
```
sentiment4_df_pred = nb_clf.predict(test_X)
```

[260]:
```
print(metrics.classification_report(test_y, sentiment4_df_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       435
           1       1.00      0.99      1.00       390

    accuracy                           1.00       825
   macro avg       1.00      1.00      1.00       825
weighted avg       1.00      1.00      1.00       825
```

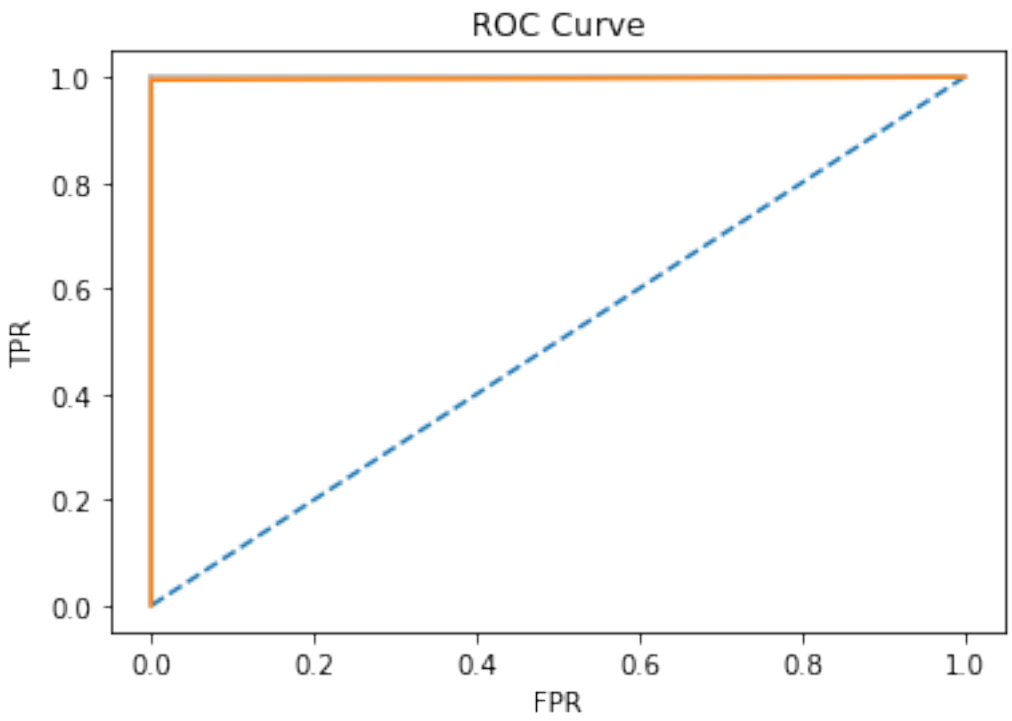
[261]:
```
print(metrics.confusion_matrix(test_y, sentiment4_df_pred))
```

```
[[435   0]
 [  2 388]]
```

[262]:
```
print(metrics.roc_auc_score(test_y,sentiment4_df_pred))
```

```
0.9974358974358974
```

[263]:
```
# Plot ROC curve
fpr, tpr, thresholds = roc_curve(test_y,sentiment4_df_pred)
plt.clf()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.plot(fpr,tpr)
plt.show()
```



[264]:
```
#Using Term Frequency-Inverse Document Frequency(TF-IDF) Model
tfidf_vectorizer = TfidfVectorizer(analyzer = stemmed_words,max_features=2500)
feature_vector = tfidf_vectorizer.fit(sentiment_df.Text)
sentiment_df_features4 = tfidf_vectorizer.transform(sentiment_df.Text)
features4 = feature_vector.get_feature_names()
```

[265]:
```
train_X1,test_X1,train_y1,test_y1 =␣
↪train_test_split(sentiment_df_features4,sentiment_df.Sentiment,test_size=0.
↪3,random_state=42)
```

[266]:
```
nb1_clf = GaussianNB()
nb1_clf.fit(train_X1.toarray(),train_y1)
```

[266]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

[267]:
```
sentiment_tfidf_pred = nb1_clf.predict(test_X1.toarray())
```

[268]:
```
print(metrics.classification_report(test_y1,sentiment_tfidf_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.67      0.70       435
           1       0.66      0.73      0.69       390
```

```
    accuracy                           0.70       825
   macro avg       0.70      0.70      0.70       825
weighted avg       0.70      0.70      0.70       825
```

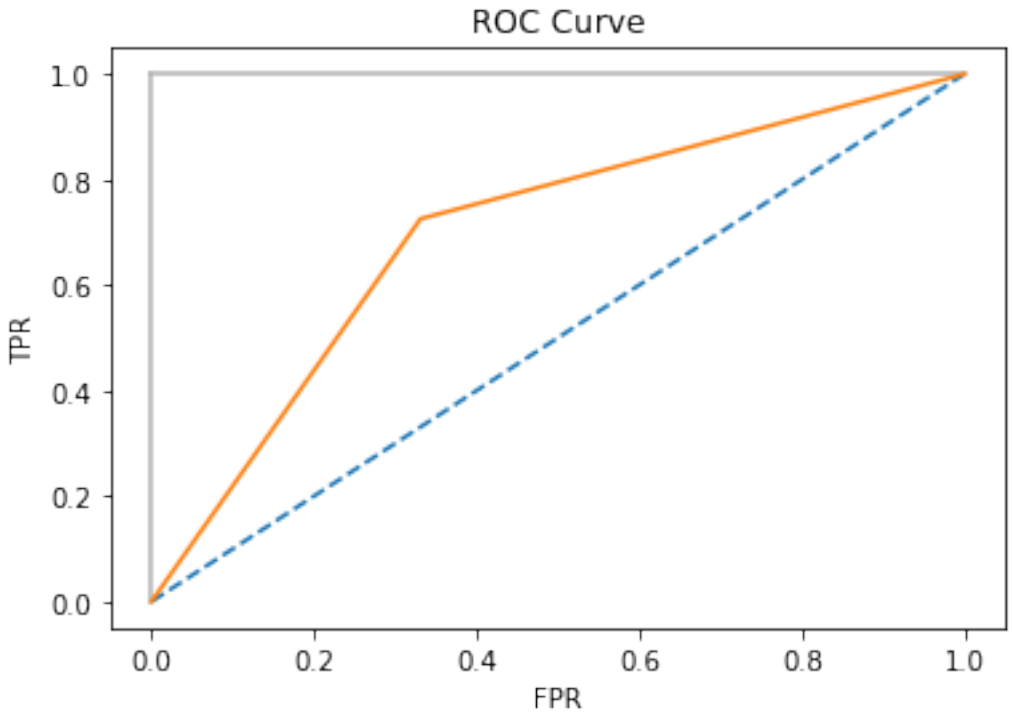
[269]:
```
print(metrics.confusion_matrix(test_y1, sentiment_tfidf_pred))
```

```
[[291 144]
 [107 283]]
```

[270]:
```
print(metrics.roc_auc_score(test_y1,sentiment_tfidf_pred))
```

```
0.6973032714412025
```

[271]:
```
# Plot ROC curve
fpr, tpr, thresholds = roc_curve(test_y1,sentiment_tfidf_pred)
plt.clf()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.plot(fpr,tpr)
plt.show()
```



[272]:
```
# using n-grams
stemmer = PorterStemmer()
def get_stemmed_tokens(doc):
    #Tokanize the documents to words
    all_tokens = [word for word in word_tokenize(doc)]
    clean_tokens = []
    # remove the all characters other than alphabets. It takes a regex for␣
 ↪matching.
    for each_token in all_tokens:
        if re.search('[a-zA-Z]', each_token):
            clean_tokens.append(each_token)
    # Stem the words
    stemmed_tokens = [stemmer.stem(t) for t in clean_tokens]
    return stemmed_tokens
```

[284]:
```
tfidf_vectorizer =␣
 ↪TfidfVectorizer(max_features=500,stop_words='english',tokenizer=get_stemmed_tokens,ngram_ra
feature_vector = tfidf_vectorizer.fit( sentiment_df.Text )
sentiment_df_features5 = tfidf_vectorizer.transform(sentiment_df.Text)
features5 = feature_vector.get_feature_names()
```

[285]:
```
train_X2, test_X2, train_y2, test_y2 = train_test_split(␣
 ↪sentiment_df_features5,sentiment_df.Sentiment,test_size = 0.3,random_state =␣
 ↪42 )
```

[286]:
```
nb2_clf = BernoulliNB()
```

[287]:
```
nb2_clf.fit(train_X2.toarray(), train_y2)
```

[287]:
```
BernoulliNB(alpha=1.0, binarize=0.0, class_prior=None, fit_prior=True)
```

[288]:
```
test_X2.shape
```

[288]:
```
(825, 500)
```

[289]:
```
sentiment_ngram_pred = nb2_clf.predict( test_X2.toarray() )
```

[290]:
```
print( metrics.classification_report(test_y2, sentiment_ngram_pred) )
```

```
              precision    recall  f1-score   support

           0       0.82      0.71      0.76       435
           1       0.72      0.83      0.77       390

    accuracy                           0.77       825
   macro avg       0.77      0.77      0.77       825
```

```
weighted avg       0.77      0.77      0.77       825
```

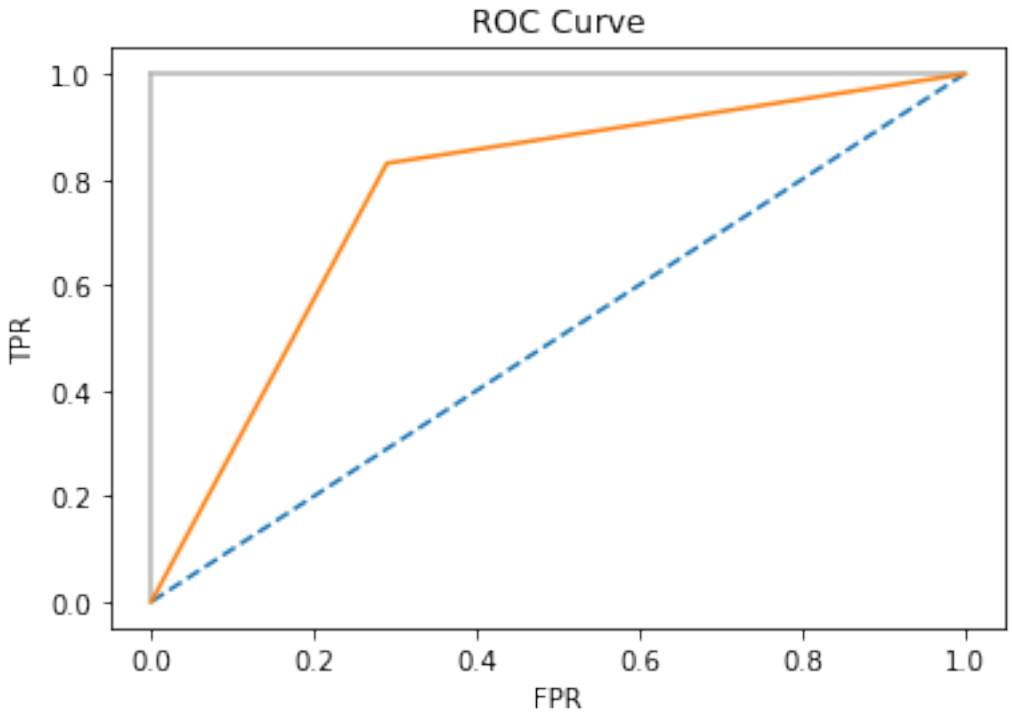
[291]:
```
print(metrics.confusion_matrix(test_y2, sentiment_ngram_pred))
```

```
[[309 126]
 [ 66 324]]
```

[292]:
```
print(metrics.roc_auc_score(test_y2, sentiment_ngram_pred))
```

```
0.7705570291777188
```

[293]:
```
# Plot ROC curve
fpr, tpr, thresholds = roc_curve(test_y2, sentiment_ngram_pred)
plt.clf()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.plot([0, 1], ls="--")
plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
plt.plot(fpr,tpr)
plt.show()
```



[ ]: