Amazon Fine Food Reviews Analysis

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews (https://www.kaggle.com/snap/amazon-fine-food-reviews)

EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/ (https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/)

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454
Number of users: 256,059
Number of products: 74,258
Timespan: Oct 1999 - Oct 2012
Number of Attributes/Columns in data: 10

Attribute Information:

1. Id
2. ProductId - unique identifier for the product
3. UserId - unqiue identifier for the user
4. ProfileName
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
7. Score - rating between 1 and 5
8. Time - timestamp for the review
9. Summary - brief summary of the review
10. Text - text of the review

Objective:

Given a review, determine whether the review is positive (rating of 4 or 5) or negative (rating of 1 or 2).

[Q] How to determine if a review is positive or negative?

[Ans] We could use Score/Rating. A rating of 4 or 5 can be cosnidered as a positive review. A rating of 1 or 2 can be considered as negative one. A review of rating 3 is considered nuetral and such reviews are ignored from our analysis. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

[1]. Reading Data

[1.1] Loading the data

The dataset is available in two forms

1. .csv file
2. SQLite Database

In order to load the data, We have used the SQLITE dataset as it is easier to query the data and visualise the data efficiently.

Here as we only want to get the global sentiment of the recommendations (positive or negative), we will purposefully ignore all Scores equal to 3. If the score is above 3, then the recommendation wil be set to "positive". Otherwise, it will be set to "negative".

In [19]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer
from nltk.stem.snowball import SnowballStemmer

from bs4 import BeautifulSoup
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os
```

In [20]:

```
# using SQLite Table to read data.
con = sqlite3.connect('E:/appliedaiacourse/assignments/dblite/database.sqli
te')

# filtering only positive and negative reviews i.e.
# not taking into consideration those reviews with Score=3
# SELECT * FROM Reviews WHERE Score != 3 LIMIT 500000, will give top 500000
data points
# you can change the number to any other number based on your computing pow
er

# filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score !
= 3 LIMIT 500000""", con)
# for tsne assignment you can take 5k data points

#filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score !=
3 LIMIT 5000""", con)
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score !=
 3 """, con)

# Give reviews with Score>3 a positive rating(1), and reviews with a score<
3 a negative rating(0).
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)
```

```
Number of data points in our data (525814, 10)
```

Out[20]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulne
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1

In [21]:

```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [22]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[22]:

	UserId	ProductId	ProfileName	Time	Score	Text	COU
0	#oc-R115TNMSPFT9I7	B005ZBZLT4	Breyton	1331510400	2	Overall its just OK when considering the price...	2
1	#oc-R11D9D7SHXIJB9	B005HG9ESG	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B005ZBZLT4	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ESG	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBEV0	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

In [23]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[23]:

	UserId	ProductId	ProfileName	Time	Score	Text	COU
80638	AZY10LLTJ71NX	B001ATMQK2	undertheshrine "undertheshrine"	1296691200	5	I bought this 6 pack because for the price tha...	5

```
In [24]: display['COUNT(*)'].sum()

Out[24]: 393063
```

[2] Exploratory Data Analysis

[2.1] Data Cleaning: Deduplication

It is observed (as shown in the table below) that the reviews data had many duplicate entries. Hence it was necessary to remove duplicates in order to get unbiased results for the analysis of the data. Following is an example:

In [25]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[25]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpful
0	78445	B000HDL1RQ	AR5J8UI46CURR	Geetha Krishnan	2	2
1	138317	B000HDOPYC	AR5J8UI46CURR	Geetha Krishnan	2	2
2	138277	B000HDOPYM	AR5J8UI46CURR	Geetha Krishnan	2	2
3	73791	B000HDOPZG	AR5J8UI46CURR	Geetha Krishnan	2	2
4	155049	B000PAQ75C	AR5J8UI46CURR	Geetha Krishnan	2	2

As it can be seen above that same user has multiple reviews with same values for HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary and Text and on doing analysis it was found that

ProductId=B000HDOPZG was Loacker Quadratini Vanilla Wafer Cookies, 8.82-Ounce Packages (Pack of 8)

ProductId=B000HDL1RQ was Loacker Quadratini Lemon Wafer Cookies, 8.82-Ounce Packages (Pack of 8) and so on

It was inferred after analysis that reviews with same parameters other than ProductId belonged to the same product just having different flavour or quantity. Hence in order to reduce redundancy it was decided to eliminate the rows having same parameters.

The method used for the same was that we first sort the data according to ProductId and then just keep the first similar product review and delelte the others. for eg. in the above just the review for ProductId=B000HDL1RQ remains. This method ensures that there is only one representative for each product and deduplication without sorting would lead to possibility of different representatives still existing for the same product.

In [26]:

```
#Sorting data according to ProductId in ascending order
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True,
inplace=False, kind='quicksort', na_position='last')
```

In [27]:

```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Te
xt"}, keep='first', inplace=False)
final.shape
```

Out[27]:

```
(364173, 10)
```

In [28]:

```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[28]:

```
69.25890143662969
```

Observation:- It was also seen that in two rows given below the value of HelpfulnessNumerator is greater than HelpfulnessDenominator which is not practically possible hence these two rows too are removed from calcualtions

In [29]:

```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[29]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpful
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	1
1	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	2

In [30]:

```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

[3] Preprocessing

[3.1]. Preprocessing Review Text

Now that we have finished deduplication our data requires some preprocessing before we go on further with analysis and making the prediction model.

Hence in the Preprocessing phase we do the following in the order below:-

1. Begin by removing the html tags
2. Remove any punctuations or limited set of special characters like , or . or # etc.
3. Check if the word is made up of english letters and is not alpha-numeric
4. Check to see if the length of the word is greater than 2 (as it was researched that there is no adjective in 2-letters)
5. Convert the word to lowercase
6. Remove Stopwords
7. Finally Snowball Stemming the word (it was obsereved to be better than Porter Stemming)

After which we collect the words used to describe positive and negative reviews

In [15]:

```
# printing some random reviews
sent_0 = final['Text'].values[0]
print(sent_0)
print("="*50)

sent_1000 = final['Text'].values[1000]
print(sent_1000)
print("="*50)

sent_1500 = final['Text'].values[1500]
print(sent_1500)
print("="*50)

sent_4900 = final['Text'].values[4900]
print(sent_4900)
print("="*50)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /
>http://www.amazon.com/VICTOR-FLY-MAGNET-BAIT-REFILL/dp/B00004RBDY<br /><br
/>The Victor M380 and M502 traps are unreal, of course -- total fly genocid
e. Pretty stinky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious these
chips are.  The best thing was that there were a lot of "brown" chips in th
e bsg (my favorite), so I bought some more through amazon and shared with f
amily and friends.  I am a little disappointed that there are not, so far,
very many brown chips in these bags, but the flavor is still very good.  I
like them better than the yogurt and green onion flavor because they do not
seem to be as salty, and the onion flavor is better.  If you haven't eaten
Kettle chips before, I recommend that you try a bag before buying bulk.  Th
ey are thicker and crunchier than Lays but just as fresh out of the bag.
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what they we
re ordering; the other wants crispy cookies.  Hey, I'm sorry; but these rev
iews do nobody any good beyond reminding us to look  before ordering.<br />
<br />These are chocolate-oatmeal cookies.  If you don't like that combinat
ion, don't order this type of cookie.  I find the combo quite nice, really.
The oatmeal sort of "calms" the rich chocolate flavor and gives the cookie
sort of a coconut-type consistency.  Now let's also remember that tastes di
ffer; so, I've given my opinion.<br /><br />Then, these are soft, chewy coo
kies -- as advertised.  They are not "crispy" cookies, or the blurb would s
ay "crispy," rather than "chewy."  I happen to like raw cookie dough; howev
er, I don't see where these taste like raw cookie dough.  Both are soft, ho
wever, so is this the confusion?  And, yes, they stick together.  Soft cook
ies tend to do that.  They aren't individually wrapped, which would add to
the cost.  Oh yeah, chocolate chip cookies tend to be somewhat sweet.<br />
<br />So, if you want something hard and crisp, I suggest Nabiso's Ginger S
naps.  If you want a cookie that's soft, chewy and tastes like a combinatio
n of chocolate and oatmeal, give these a try.  I'm here to place my second
order.
==================================================
love to order my coffee on amazon.  easy and shows up quickly.<br />This k
cup is great coffee.  dcaf is very good as well
==================================================
```

In [16]:

```
# remove urls from text python: https://stackoverflow.com/a/40823105/408403
9
sent_0 = re.sub(r"http\S+", "", sent_0)
sent_1000 = re.sub(r"http\S+", "", sent_1000)
sent_150 = re.sub(r"http\S+", "", sent_1500)
sent_4900 = re.sub(r"http\S+", "", sent_4900)

print(sent_0)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /
> /><br />The Victor M380 and M502 traps are unreal, of course -- total fly
genocide. Pretty stinky, but only right nearby.
```

In [17]:

```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how-to-remove-all-tags-from-an-element
from bs4 import BeautifulSoup

soup = BeautifulSoup(sent_0, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1000, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1500, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_4900, 'lxml')
text = soup.get_text()
print(text)
```

```
Why is this $[...] when the same product is available for $[...] here? />Th
e Victor M380 and M502 traps are unreal, of course -- total fly genocide. P
retty stinky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious these
chips are.  The best thing was that there were a lot of "brown" chips in th
e bsg (my favorite), so I bought some more through amazon and shared with f
amily and friends.  I am a little disappointed that there are not, so far,
very many brown chips in these bags, but the flavor is still very good.  I
like them better than the yogurt and green onion flavor because they do not
seem to be as salty, and the onion flavor is better.  If you haven't eaten
Kettle chips before, I recommend that you try a bag before buying bulk.  Th
ey are thicker and crunchier than Lays but just as fresh out of the bag.
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what they we
re ordering; the other wants crispy cookies.  Hey, I'm sorry; but these rev
iews do nobody any good beyond reminding us to look  before ordering.These
are chocolate-oatmeal cookies.  If you don't like that combination, don't o
rder this type of cookie.  I find the combo quite nice, really.  The oatmea
l sort of "calms" the rich chocolate flavor and gives the cookie sort of a
coconut-type consistency.  Now let's also remember that tastes differ; so,
I've given my opinion.Then, these are soft, chewy cookies -- as advertised.
They are not "crispy" cookies, or the blurb would say "crispy," rather than
"chewy."  I happen to like raw cookie dough; however, I don't see where the
se taste like raw cookie dough.  Both are soft, however, so is this the con
fusion?  And, yes, they stick together.  Soft cookies tend to do that.  The
y aren't individually wrapped, which would add to the cost.  Oh yeah, choco
late chip cookies tend to be somewhat sweet.So, if you want something hard
and crisp, I suggest Nabiso's Ginger Snaps.  If you want a cookie that's so
ft, chewy and tastes like a combination of chocolate and oatmeal, give thes
e a try.  I'm here to place my second order.
==================================================
love to order my coffee on amazon.  easy and shows up quickly.This k cup is
great coffee.  dcaf is very good as well
```

In [18]:

```python
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [19]:
```
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
Wow.  So far, two two-star reviews.  One obviously had no idea what they we
re ordering; the other wants crispy cookies.  Hey, I am sorry; but these re
views do nobody any good beyond reminding us to look  before ordering.<br /
><br />These are chocolate-oatmeal cookies.  If you do not like that combin
ation, do not order this type of cookie.  I find the combo quite nice, real
ly.  The oatmeal sort of "calms" the rich chocolate flavor and gives the co
okie sort of a coconut-type consistency.  Now let is also remember that tas
tes differ; so, I have given my opinion.<br /><br />Then, these are soft, c
hewy cookies -- as advertised.  They are not "crispy" cookies, or the blurb
would say "crispy," rather than "chewy."  I happen to like raw cookie doug
h; however, I do not see where these taste like raw cookie dough.  Both are
soft, however, so is this the confusion?  And, yes, they stick together.  S
oft cookies tend to do that.  They are not individually wrapped, which woul
d add to the cost.  Oh yeah, chocolate chip cookies tend to be somewhat swe
et.<br /><br />So, if you want something hard and crisp, I suggest Nabiso i
s Ginger Snaps.  If you want a cookie that is soft, chewy and tastes like a
combination of chocolate and oatmeal, give these a try.  I am here to place
my second order.
==================================================
```

In [20]:
```
#remove words with numbers python: https://stackoverflow.com/a/18082370/408
4039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /
> /><br />The Victor  and  traps are unreal, of course -- total fly genocid
e. Pretty stinky, but only right nearby.
```

In [21]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
print(sent_1500)
```

```
Wow So far two two star reviews One obviously had no idea what they were or
dering the other wants crispy cookies Hey I am sorry but these reviews do n
obody any good beyond reminding us to look before ordering br br These are
chocolate oatmeal cookies If you do not like that combination do not order
this type of cookie I find the combo quite nice really The oatmeal sort of
calms the rich chocolate flavor and gives the cookie sort of a coconut type
consistency Now let is also remember that tastes differ so I have given my
opinion br br Then these are soft chewy cookies as advertised They are not
crispy cookies or the blurb would say crispy rather than chewy I happen to
like raw cookie dough however I do not see where these taste like raw cooki
e dough Both are soft however so is this the confusion And yes they stick t
ogether Soft cookies tend to do that They are not individually wrapped whic
h would add to the cost Oh yeah chocolate chip cookies tend to be somewhat
sweet br br So if you want something hard and crisp I suggest Nabiso is Gin
ger Snaps If you want a cookie that is soft chewy and tastes like a combina
tion of chocolate and oatmeal give these a try I am here to place my second
order
```

In [22]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# <br /><br /> ==> after the above steps, we are getting "br br"
# we are including them into stop words list
# instead of <br /> if we have <br/> these tags would have revmoved in the
 1st step

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours'
, 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have'
, 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'be
cause', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'int
o', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on'
, 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so',
'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "sho
uld've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'did
n', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "sh
ouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

In [23]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(final['Text'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() no
t in stopwords)
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 4986/4986 [00:02<00:00, 2257.00it/s]
```

In [24]:
```
preprocessed_reviews[1500]
print(type(preprocessed_reviews))
print(final.shape)
```

```
<class 'list'>
(4986, 10)
```

[3.2] Preprocessing Review Summary

In [25]:
```
## Similartly you can do preprocessing for review summary also.
```

[4] Featurization

[4.1] BAG OF WORDS

In [24]:
```
#BoW
count_vect = CountVectorizer() #in scikit-learn
count_vect.fit(preprocessed_reviews)
print("some feature names ", count_vect.get_feature_names()[:10])
print('='*50)

final_counts = count_vect.transform(preprocessed_reviews)
print("the type of count vectorizer ",type(final_counts))
print("the shape of out text BOW vectorizer ",final_counts.get_shape())
print("the number of unique words ", final_counts.get_shape()[1])
```

```
some feature names  ['aa', 'aahhhs', 'aback', 'abandon', 'abates', 'abbot
t', 'abby', 'abdominal', 'abiding', 'ability']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (4986, 12997)
the number of unique words  12997
```

[4.2] Bi-Grams and n-Grams.

In [26]:

```
#bi-gram, tri-gram and n-gram

#removing stop words like "not" should be avoided before building n-grams
# count_vect = CountVectorizer(ngram_range=(1,2))
# please do read the CountVectorizer documentation http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

# you can choose these numebrs min_df=10, max_features=5000, of your choice
count_vect = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
final_bigram_counts = count_vect.fit_transform(preprocessed_reviews)
print("the type of count vectorizer ",type(final_bigram_counts))
print("the shape of out text BOW vectorizer ",final_bigram_counts.get_shape())
print("the number of unique words including both unigrams and bigrams ", final_bigram_counts.get_shape()[1])
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (4986, 3144)
the number of unique words including both unigrams and bigrams  3144
```

[4.3] TF-IDF

In [27]:

```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
tf_idf_vect.fit(preprocessed_reviews)
print("some sample features(unique words in the corpus)",tf_idf_vect.get_feature_names()[0:10])
print('='*50)

final_tf_idf = tf_idf_vect.transform(preprocessed_reviews)
print("the type of count vectorizer ",type(final_tf_idf))
print("the shape of out text TFIDF vectorizer ",final_tf_idf.get_shape())
print("the number of unique words including both unigrams and bigrams ", final_tf_idf.get_shape()[1])
```

```
some sample features(unique words in the corpus) ['ability', 'able', 'able
find', 'able get', 'absolute', 'absolutely', 'absolutely delicious', 'absol
utely love', 'absolutely no', 'according']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text TFIDF vectorizer  (4986, 3144)
the number of unique words including both unigrams and bigrams  3144
```

[4.4] Word2Vec

In [28]:

```
# Train your own Word2Vec model using your own text corpus
i=0
list_of_sentance=[]
for sentance in preprocessed_reviews:
    list_of_sentance.append(sentance.split())
```

In [29]:

```
# Using Google News Word2Vectors

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file wich contains a dict ,
# and it contains all our courpus words as keys and  model[word] as values
# To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
# it's 1.9GB in size.


# http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17SRFA
zZPY
# you can comment this whole cell
# or change these varible according to your need

is_your_ram_gt_16g=False
want_to_use_google_w2v = False
want_to_train_w2v = True

if want_to_train_w2v:
    # min_count = 5 considers only words that occured atleast 5 times
    w2v_model=Word2Vec(list_of_sentance,min_count=5,size=50, workers=4)
    print(w2v_model.wv.most_similar('great'))
    print('='*50)
    print(w2v_model.wv.most_similar('worst'))

elif want_to_use_google_w2v and is_your_ram_gt_16g:
    if os.path.isfile('GoogleNews-vectors-negative300.bin'):
        w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors-neg
ative300.bin', binary=True)
        print(w2v_model.wv.most_similar('great'))
        print(w2v_model.wv.most_similar('worst'))
    else:
        print("you don't have gogole's word2vec file, keep want_to_train_w2
v = True, to train your own w2v ")
```

```
[('excellent', 0.9934042692184448), ('looking', 0.9928814768791199), ('heal
thy', 0.9919042587280273), ('overall', 0.9917173385620117), ('worth', 0.991
6903972625732), ('terrific', 0.9916651248931885), ('either', 0.991615474224
0906), ('amazing', 0.9915229082107544), ('fantastic', 0.9915140867233276),
('anything', 0.9914295673370361)]
==================================================
[('results', 0.9992072582244873), ('tomatoes', 0.9991599917411804), ('c',
0.9991344213485718), ('become', 0.9991328716278076), ('wife', 0.99911159276
96228), ('enjoyed', 0.9990763664245605), ('remember', 0.9990705251693726),
('beef', 0.9990602731704712), ('tassimo', 0.9990425109863281), ('superior',
0.9990386962890625)]
```

```
In [30]: w2v_words = list(w2v_model.wv.vocab)
         print("number of words that occured minimum 5 times ",len(w2v_words))
         print("sample words ", w2v_words[0:50])
```

```
number of words that occured minimum 5 times  3817
sample words  ['product', 'available', 'course', 'total', 'pretty', 'stink
y', 'right', 'nearby', 'used', 'ca', 'not', 'beat', 'great', 'received', 's
hipment', 'could', 'hardly', 'wait', 'try', 'love', 'call', 'instead', 'rem
oved', 'easily', 'daughter', 'designed', 'printed', 'use', 'car', 'window
s', 'beautifully', 'shop', 'program', 'going', 'lot', 'fun', 'everywhere',
'like', 'tv', 'computer', 'really', 'good', 'idea', 'final', 'outstanding',
'window', 'everybody', 'asks', 'bought', 'made']
```

[4.4.1] Converting text into vectors using Avg W2V, TFIDF-W2V

[4.4.1.1] Avg W2v

```
In [31]: # average Word2Vec
         # compute average word2vec for each review.
         sent_vectors = []; # the avg-w2v for each sentence/review is stored in this
         list
         for sent in tqdm(list_of_sentance): # for each review/sentence
             sent_vec = np.zeros(50) # as word vectors are of zero length 50, you mi
         ght need to change this to 300 if you use google's w2v
             cnt_words =0; # num of words with a valid vector in the sentence/review
             for word in sent: # for each word in a review/sentence
                 if word in w2v_words:
                     vec = w2v_model.wv[word]
                     sent_vec += vec
                     cnt_words += 1
             if cnt_words != 0:
                 sent_vec /= cnt_words
             sent_vectors.append(sent_vec)
         print(len(sent_vectors))
         print(len(sent_vectors[0]))
```

```
100%|██████████| 4986/4986 [00:05<00:00, 910.13it/s]

4986
50
```

[4.4.1.2] TFIDF weighted W2v

```
In [32]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
         model = TfidfVectorizer()
         tf_idf_matrix = model.fit_transform(preprocessed_reviews)
         # we are converting a dictionary with word as a key, and the idf as a value
         dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))
```

```
In [33]: # TF-IDF weighted Word2Vec
         tfidf_feat = model.get_feature_names() # tfidf words/col-names
         # final_tf_idf is the sparse matrix with row= sentence, col=word and cell_v
         al = tfidf

         tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored
         in this list
         row=0;
         for sent in tqdm(list_of_sentance): # for each review/sentence
             sent_vec = np.zeros(50) # as word vectors are of zero length
             weight_sum =0; # num of words with a valid vector in the sentence/revie
         w
             for word in sent: # for each word in a review/sentence
                 if word in w2v_words and word in tfidf_feat:
                     vec = w2v_model.wv[word]
         #             tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
                     # to reduce the computation we are
                     # dictionary[word] = idf value of word in whole courpus
                     # sent.count(word) = tf valeus of word in this review
                     tf_idf = dictionary[word]*(sent.count(word)/len(sent))
                     sent_vec += (vec * tf_idf)
                     weight_sum += tf_idf
             if weight_sum != 0:
                 sent_vec /= weight_sum
             tfidf_sent_vectors.append(sent_vec)
             row += 1
```

```
100%|██████████| 4986/4986 [00:26<00:00, 191.22it/s]
```

[5] Assignment 4: Apply Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**

   - SET 1:Review text, preprocessed one converted into vectors using (BOW)
   - SET 2:Review text, preprocessed one converted into vectors using (TFIDF)

2. **The hyper paramter tuning(find best Alpha)**

   - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
   - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
   - Find the best hyper paramter using k-fold cross validation or simple cross validation data
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Feature importance**

   - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names

4. **Feature engineering**

   - To increase the performance of your model, you can also experiment with with feature engineering like :
     - Taking length of reviews as another feature.
     - Considering some features from review summary as well.

5. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
     Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
     Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.
     (https://seaborn.pydata.org/generated/seaborn.heatmap.html)
     (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

   (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

   (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

6. **Conclusion** (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

- You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

Applying Multinomial Naive Bayes

[5.1] Applying Naive Bayes on BOW, SET 1

In [0]:

```
# Please write all the code with proper documentation
```

In [2]:

```
# the required imports
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import classification_report,accuracy_score,confusion_
matrix
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from json import dump,loads
import pandas as pd
import numpy as np
import math
import os
import time
import enum
import scipy
import csv

class ratiodatasplit(enum.Enum):
    high=0.2
    medium = 0.3
    low = 0.4

class multiNaiveBayes:
    def __init__(self):
        self.Xdata=[]
        self.Xdatavect = pd.DataFrame()
        self.ydata=pd.DataFrame()
        self.xtrain=pd.DataFrame()
        self.xtest=pd.DataFrame()
        self.xval=pd.DataFrame()
        self.ytrain= pd.Series([])
        self.ytest= pd.Series([])
        self.yval= pd.Series([])
        self.mnb_clf = None
        self.NBayes_alpha = []
        self.yprdprobatrn = []
        self.yprdprobaval = []
        self.yprdprobatest = []
        self.ytrn_predprob_actclf = []
        self.ytst_predprob_actclf = []
        self.rocaucscoretrn = []
        self.rocaucscoreval = []
        self.rocaucscoretest = []
        self.predicted = []
        self.test_predict = []
        self.accuracy_score_val = []
        self.accuracy_score_test = []
        self.clasify_report = []
        self.confsnmtxytstpred = {}
        self.roc_curve_test = {}
        self.clasify_params = {}
        self.graph_params = {}
        self.outputdir = None
        self.opdataitem = {}
        self.opdatajson = {}
        self.count_vect = None
        self.tf_idf_vect = None

    def multNBClasify(self):
        self.mnb_clf = MultinomialNB()
        return self.mnb_clf

    def getNBClassifier(self):
        return self.mnb_clf

    @property
    def mnb_clf(self):
        return self._mnb_clf

    @mnb_clf.setter
    def mnb_clf(self,new_mnbclf):
        self._mnb_clf = new_mnbclf

    @property
    def Xdata(self):
        return self._Xdata

    @Xdata.setter
    def Xdata(self,new_Xdata):
        self._Xdata = new_Xdata


    @property
    def Xdatavect(self):
        return self._Xdatavect

    @Xdatavect.setter
    def Xdatavect(self,new_Xdatavect):
        self._Xdatavect = new_Xdatavect

    @property
    def ydata(self):
        return self._ydata

    @ydata.setter
    def ydata(self,new_ydata):
        self._ydata = new_ydata


    @property
    def xtrain(self):
        return self._xtrain

    @xtrain.setter
    def xtrain(self,new_xtrain):
        self._xtrain = new_xtrain


    @property
    def xtest(self):
        return self._xtest

    @xtest.setter
    def xtest(self,new_xtest):
        self._xtest = new_xtest

    @property
    def xval(self):
        return self._xval

    @xval.setter
    def xval(self,new_xval):
        self._xval = new_xval

    @property
    def ytrain(self):
        return self._ytrain

    @ytrain.setter
    def ytrain(self,new_ytrain):
        self._ytrain = new_ytrain

    @property
    def ytest(self):
        return self._ytest

    @ytest.setter
    def ytest(self,new_ytest):
        self._ytest = new_ytest

    @property
    def yval(self):
        return self._yval

    @yval.setter
    def yval(self,new_yval):
        self._yval = new_yval

    @property
    def yprdprobatrn(self):
        return self._yprdprobatrn

    @yprdprobatrn.setter
    def yprdprobatrn(self,new_yprdprobatrn):
        self._yprdprobatrn = new_yprdprobatrn

    @property
    def yprdprobaval (self):
        return self._yprdprobaval

    @yprdprobaval.setter
    def yprdprobaval (self,new_yprdprobaval):
        self._yprdprobaval = new_yprdprobaval


    @property
    def yprdprobatest (self):
        return self._yprdprobatest

    @yprdprobatest.setter
    def yprdprobatest (self,new_yprdprobatest):
        self._yprdprobatest = new_yprdprobatest


    @property
    def ytrn_predprob_actclf (self):
        return self._ytrn_predprob_actclf

    @ytrn_predprob_actclf.setter
    def ytrn_predprob_actclf (self,new_ytrn_predprob_actclf):
        self._ytrn_predprob_actclf = new_ytrn_predprob_actclf

    @property
    def NBayes_alpha (self):
        return self._NBayes_alpha

    @NBayes_alpha.setter
    def NBayes_alpha (self,new_NBayes_alpha):
        self._NBayes_alpha = new_NBayes_alpha


    @property
    def outputdir (self):
        return self._outputdir

    @outputdir.setter
    def outputdir (self,new_outputdir):
        self._outputdir = new_outputdir


    def setalphaparm(self,prmval):
        params = {'alpha':prmval}
        (self.mnb_clf).set_params(**params)
        return self.mnb_clf

    def mnb_fitdata(self):
        mnb_clf.fit(self.xdata,self.ydata)
        return self.mnb_clf

    def mnb_predict(self):
        predicted = mnb_clf.predict(self.xtest)
        return [predicted,mb_clf]

    def hyperparamtuning(self,hyperparam,measure,cvfold=5,vbose=100,njob=1
):


        # set the parameter values for hyertuning
        param_grid = {'alpha':hyperparam}

        #initialize the classifier
        grdsch_clf = self.getNBClassifier()
        grdschcv = GridSearchCV(grdsch_clf, param_grid,scoring=measure, cv
= cvfold, verbose=vbose, n_jobs=njob)

        #fit the data with classifier
        grdschcv.fit(self.xtrain,self.ytrain)
        return [grdschcv.best_score_,grdschcv.best_params_,grdschcv]

    def splitdatasets(self,splitratio,proportion,standardise,randomseed=42
):
        #split into train and test sets (80/20)

        if standardise :
            if scipy.sparse.issparse(self.Xdata) :
                X1data = self.Xdata
                self.Xdata = X1data.todense()
        """
        print((self.Xdata).shape,(self.ydata).shape)
        #print((self.Xdata),(self.ydata))
        print(proportion)
        """

        if proportion :
            X_train8, X_test8, y_train8, y_test8 = train_test_split(self.Xd
ata, self.ydata, stratify=self.ydata,test_size=splitratio.value, random_sta
te=randomseed)
            X_trn8, X_val8, y_trn8, y_val8   = train_test_split(X_train8, y
_train8,stratify=y_train8, test_size=splitratio.value, random_state=randoms
eed)
        else:
            X_train8, X_test8, y_train8, y_test8 = train_test_split(self.Xd
ata, self.ydata, test_size=splitratio.value, random_state=randomseed)
            X_trn8, X_val8, y_trn8, y_val8   = train_test_split(X_train8, y
_train8, test_size=splitratio.value, random_state=randomseed)

        # statndardize the data
        if standardise:
            scaler = StandardScaler()
            train8_scaled = scaler.fit_transform(X_trn8)
            test8_scaled = scaler.transform(X_test8)
            v8_scaled = scaler.transform(X_val8)
            self.xtrain = train8_scaled
            self.xtest = test8_scaled
            self.xval = v8_scaled
            self.ytrain = y_trn8
            self.ytest = y_test8
            self.yval = y_val8
        else:
            self.xtrain = X_trn8
            self.xtest = X_test8
            self.xval = X_val8
            self.ytrain = y_trn8
            self.ytest = y_test8
            self.yval = y_val8
            self.xtest = X_test8

    def BOWVectorizer(self):
        #BoW
        self.count_vect = CountVectorizer(max_features=1000) #in scikit-lea
```

```
rn
        self.count_vect.fit(self.xtrain)
        print("some feature names ", self.count_vect.get_feature_names()[:1
0])
        print('='*50)

        self.xtrain = self.count_vect.transform(self.xtrain)
        self.xtest = self.count_vect.transform(self.xtest)
        self.xval = self.count_vect.transform(self.xval)
        print("the type of count vectorizer ",type(self.xtrain))
        print("the shape of out text BOW vectorizer ",self.xtrain.get_shape
())
        print("the number of unique words ", self.xtrain.get_shape()[1])


    def tfIdfVectorizer(self):
        self.tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10,)
        self.tf_idf_vect.fit(self.xtrain)
        print("some sample features(unique words in the corpus)",self.tf_id
f_vect.get_feature_names()[0:10])
        print('='*50)

        self.xtrain = self.tf_idf_vect.transform(self.xtrain)
        self.xtest = self.tf_idf_vect.transform(self.xtest)
        self.xval  = self.tf_idf_vect.transform(self.xval)
        print("the type of count vectorizer ",type(self.xtrain))
        print("the shape of out text TFIDF vectorizer ",self.xtrain.get_sha
pe())
        print("the number of unique words including both unigrams and bigra
ms ", self.xtrain.get_shape()[1])

    def tstloop(self,endval):
        i = 0.00000000001
        while(i <= endval):
            print(i)
            i = i*10

    def calcrocaucscore_naivebayes(self,endval):
        alpha_start = 0.00000000001
        while(alpha_start <= endval):

            # set alpha param for classifier
            self.setalphaparm(alpha_start)

            # fit the x-train model
            (self.mnb_clf).fit(self.xtrain,self.ytrain)
            self.yprdprobatrn =  (self.mnb_clf).predict_proba(self.xtrain)
[:,1]
            (self.rocaucscoretrn).append(roc_auc_score(self.ytrain,self.ypr
dprobatrn))
            print('Fitting probability generation and roc auc score generat
ion for training data complete...')

            #fit the validation model
            (self.mnb_clf).fit(self.xval,self.yval)
            self.yprdprobaval =  (self.mnb_clf).predict_proba(self.xval)[:,
1]
```

```
            (self.rocaucscoreval).append(roc_auc_score(self.yval,self.yprdprobaval))
            print('Fitting probability generation and roc auc score generation for validation data complete...')

            # predict the labels for validation
            self.predicted = (self.mnb_clf).predict(self.xval)

            # calculate accuracy_score
            self.accuracy_score_val = accuracy_score(self.yval, self.predicted)

            print('Predicting labels for training data complete...')

            #set alpha to the next value
            (self.NBayes_alpha).append(alpha_start)
            alpha_start = alpha_start * 10

        print('Function exiting...')

    def actualClasifier_naivebayes(self,parm_alpha):
        self.setalphaparm(parm_alpha)
        (self.mnb_clf).fit(self.xtest,self.ytest)

        # predict xtest labels
        self.test_predict = (self.mnb_clf).predict(self.xtest)

        #store the classifier parameters
        self.clasify_params['clfparams'] = (self.mnb_clf).get_params(deep=True)

        # calculate accuracy_score
        self.accuracy_score_test = accuracy_score(self.ytest, self.test_predict)

        # generate classification report
        #classification_report(self.ytest, self.test_predict)

        # confusion matrix for ytest
        tn, fp, fn, tp = confusion_matrix(self.ytest, self.test_predict ).ravel()
        self.confsnmtxytstpred['tn'] = tn
        self.confsnmtxytstpred['fp'] = fp
        self.confsnmtxytstpred['fn'] = fn
        self.confsnmtxytstpred['tp'] = tp

        # predict probabilites  from xtrain for roc_curve
        self.ytrn_predprob_actclf = (self.mnb_clf).predict_proba(self.xtrain)[:,1]
        fpr_trn, tpr_trn, thrshld_trn = roc_curve(self.ytrain, self.ytrn_predprob_actclf)

        # predict probabilites  from xtest for roc_curve
        self.ytst_predprob_actclf = (self.mnb_clf).predict_proba(self.xtest)[:,1]
        fpr, tpr, thrshld_test = roc_curve(self.ytest,self.ytst_predprob_actclf)
        # store the above into the dictionary
        self.roc_curve_test['fpr_trn'] = fpr_trn
        self.roc_curve_test['tpr_trn'] =  tpr_trn
        self.roc_curve_test['thrshld_trn'] = thrshld_trn
        self.roc_curve_test['fpr'] = fpr
        self.roc_curve_test['tpr'] = tpr
        self.roc_curve_test['thrshld_test'] = thrshld_test




    def exportopdatatocsv(self,name,data):
        fname = self.outputdir + "/" + name + '.csv'
        with open(fname,"w") as csvFile:
                wr=csv.writer(csvFile,quoting=csv.QUOTE_NONE,escapechar='\\
')
                wr.writerow(data)

    def exportopdatatojson(self):
        self.opdataitem['NBayes_alpha'] = self.NBayes_alpha
        self.opdataitem['yprdprobatrn'] = self.yprdprobatrn
        self.opdataitem['yprdprobaval'] = self.yprdprobaval
        self.opdataitem['yprdprobatest'] = self.yprdprobatest
        self.opdataitem['ytrn_predprob_actclf'] = self.ytrn_predprob_actclf
        self.opdataitem['ytst_predprob_actclf'] = self.ytst_predprob_actclf
        self.opdataitem['rocaucscoretrn'] = self.rocaucscoretrn
        self.opdataitem['rocaucscoreval'] = self.rocaucscoreval
        self.opdataitem['rocaucscoretest'] = self.rocaucscoretest
        self.opdataitem['predicted'] = self.predicted
        self.opdataitem['test_predict'] = self.test_predict
        self.opdatajson = {
                            'Model':'NBayesClasify',
                            'Opdata': self.opdataitem
                           }
        fname = self.outputdir + "/" + 'NBayesclasify.json'

        fp = open(fname, 'a+')
        dump(self.opdatajson, fp, indent=4)
        fp.close()
```

In [3]:

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

class drawgraphs:
    def __init__(self):
        self.graph_parameters= {}
        self.plt = None

    #self.graph_parameters['']=
    def setdefaultparm(self):
        self.Xdata=pd.DataFrame()
        self.ydatatrn=pd.DataFrame()
        self.ydataval=pd.DataFrame()
        self.graph_parameters['figsize_x']= 16
        self.graph_parameters['figsize_y']= 16
        self.graph_parameters['show_legnd']= False
        self.graph_parameters['show_grid']= True
        self.graph_title = None
        self.legnd_1x = None
        self.legnd_2 = None
        self.label_x = None
        self.label_y = None


    @property
    def Xdata(self):
        return self._Xdata

    @Xdata.setter
    def Xdata(self,new_Xdata):
        self._Xdata = new_Xdata


    @property
    def ydatatrn(self):
        return self._ydatatrn

    @ydatatrn.setter
    def ydatatrn(self,new_ydatatrn):
        self._ydatatrn = new_ydatatrn

    @property
    def ydataval(self):
        return self._ydataval

    @ydataval.setter
    def ydataval(self,new_ydataval):
        self._ydataval = new_ydataval


    @property
    def graph_title(self):
        return self._graph_title

    @graph_title.setter
    def graph_title(self,new_title):
        self._graph_title = new_title


    @property
    def legnd_1(self):
        return self._legnd_1

    @legnd_1.setter
    def legnd_1(self,new_legnd1):
        self._legnd_1 = new_legnd1


    @property
    def legnd_2(self):
        return self._legnd_2

    @legnd_2.setter
    def legnd_2(self,new_legnd2):
        self._legnd_2 = new_legnd2


    @property
    def label_x(self):
        return self._label_x

    @label_x.setter
    def label_x(self,new_lblx):
        self._label_x = new_lblx


    @property
    def label_y(self):
        return self._label_y

    @label_y.setter
    def label_y(self,new_labely):
        self._label_y = new_labely

    def rocacuscoregraph(self):
        plt.figure(figsize=(self.graph_parameters['figsize_x'],self.graph_parameters['figsize_y']))
        y1=np.asarray(self.ydatatrn)
        y1 = y1.reshape(-1,1)
        y2=np.asarray(self.ydataval)
        y2 = y2.reshape(-1,1)
        plt.plot(self.Xdata,y1, label=self.legnd_1)
        plt.plot(self.Xdata,y2, label=self.legnd_2)
        plt.xlabel(self.label_x)
        plt.ylabel(self.label_y)
        plt.title(self.graph_title)
        plt.grid(self.graph_parameters['show_grid'])

        if self.graph_parameters['show_legnd'] :
            plt.legend()
        plt.show()
    def constructgraph(self, fpr_trn, tpr_trn, fpr, tpr):
        plt.figure(figsize=(self.graph_parameters['figsize_x'],self.graph_p
arameters['figsize_y']))
        plt.plot([0,1],[0,1],'k--')
        plt.plot(fpr_trn,tpr_trn, label=self.legnd_1)
        plt.plot(fpr,tpr, label=self.legnd_2)
        plt.xlabel(self.label_x)
        plt.ylabel(self.label_y)
        plt.title(self.graph_title)
        plt.grid(self.graph_parameters['show_grid'])

        if self.graph_parameters['show_legnd'] :
            plt.legend()
        plt.show()

    def draw_table(self,data):
        colors = [["#56b5fd","w"],[ "w","#1ac3f5"]]
        table = plt.table(cellText=data,rowLabels=['Actual:\n NO','Actual:
\nYES'], colLabels=['Predicted: \n NO', 'Predicted: \n YES'], loc='center',
                          cellLoc='center',cellColours=colors, colColours=[
'Red', 'Green'],rowColours=['Yellow','Green'])

        table.set_fontsize(24)
        for i in range(0,3):
            for j in range(-1,2):
                if (i==0 and j == -1):
                    continue
                table.get_celld()[(i,j)].set_height(0.5)
                table.get_celld()[(i,j)].set_width(0.5)
                table.get_celld()[(i,j)].set_linewidth(4)
        plt.axis('off')
        plt.show()

    def draw_accscore(self,data):
        #colors = [["#56b5fd","w"]]
        table = plt.table(cellText=data,colLabels=['Validation','Test'], ro
wLabels=['Accuracy\nScore'], loc='center',
                          cellLoc='center', rowColours=['Green'],colColours
=["#56b5fd","#1ac3f5"])

        table.set_fontsize(24)
        for i in range(0,2):
            for j in range(-1,2):
                if (i==0 and j == -1):
                    continue
                table.get_celld()[(i,j)].set_height(0.5)
                table.get_celld()[(i,j)].set_width(0.8)
                table.get_celld()[(i,j)].set_linewidth(4)
        plt.axis('off')
        plt.show()


    def draw_posnegwords(self,data):
        #colors = [["#56b5fd","w"]]
        table = plt.table(cellText=data,colLabels=['Postive','Negative'], r
owLabels=['1','2','3','4','5','6','7','8','9','10'], loc='center',
                          cellLoc='center',colColours=["#56b5fd","#1ac3f5"
])

        table.set_fontsize(20)
        for i in range(0,11):
            for j in range(-1,2):
                if (i==0 and j == -1):
                    continue
                #if (i==0 and j == 2):
                    #continue
                table.get_celld()[(i,j)].set_height(0.3)
                table.get_celld()[(i,j)].set_width(0.8)
                table.get_celld()[(i,j)].set_linewidth(4)
        plt.axis('off')
        plt.show()
```

In [6]:

```
def load_data():

    import pickle

    with open ('E:/appliedaiacourse/assignments/dblite/preproc_xtrain', 'r
b') as fp:
        xtrain_preproc = pickle.load(fp)

    with open ('E:/appliedaiacourse/assignments/dblite/preproc_xtest', 'rb'
) as fp:
        xtest_preproc = pickle.load(fp)

    with open ('E:/appliedaiacourse/assignments/dblite/preproc_xval', 'rb')
as fp:
        xval_preproc = pickle.load(fp)

    with open ('E:/appliedaiacourse/assignments/dblite/ytrain', 'rb') as fp
:
        ytrain = pickle.load(fp)

    with open ('E:/appliedaiacourse/assignments/dblite/ytest', 'rb') as fp:
        ytest = pickle.load(fp)

    with open ('E:/appliedaiacourse/assignments/dblite/yval', 'rb') as fp:
        yval = pickle.load(fp)

    return [xtrain_preproc,xtest_preproc,xval_preproc,ytrain,ytest,yval]
```

In [4]:

```
#instantiate the NaiveBayes object and Multinomial Naive Bayes classifier
mnbayes = multiNaiveBayes()
mnbayes_clf = mnbayes.multNBClasify()
```

In [7]:

```
#load data
mnbayes.xtrain,mnbayes.xtest,mnbayes.xval, mnbayes.ytrain,mnbayes.ytest,mnb
ayes.yval = load_data()
```

In [8]:

```
#Bag of words vectorizer with max featues 1000
mnbayes.BOWVectorizer()
```

```
some feature names  ['abl', 'absolut', 'acid', 'across', 'actual', 'ad', 'a
dd', 'addict', 'addit', 'advertis']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (64000, 1000)
the number of unique words  1000
```

In [9]:

```
#the calssifier uses distance measure hence data has to be scaled
scaler = StandardScaler(with_mean=False)
bowtrain_scaled = scaler.fit_transform(mnbayes.xtrain.toarray())
bowtest_scaled = scaler.transform(mnbayes.xtest)
bowv_scaled = scaler.transform(mnbayes.xval)
```

In [10]:

```
# cross check data shapes
print(bowtrain_scaled.shape)
print(bowtest_scaled.shape)
print(bowv_scaled.shape)
print((mnbayes.ytrain).shape)
print((mnbayes.ytest).shape)
print((mnbayes.yval).shape)
```

```
(64000, 1000)
(20000, 1000)
(16000, 1000)
(64000,)
(20000,)
(16000,)
```

In [11]:

```
print(mnbayes.getNBClassifier())
#hyper parameter tuning to find  alpha
return_63 = mnbayes.hyperparamtuning([0.00000000001,0.0000000001,0.00000000
01,0.000000001,0.00000001,0.0000001,0.000001,0.00001,0.0001,0.001,0.01,1,10
,100,1000,10000,100000,1000000,10000000,100000000,1000000000,10000000000],
'roc_auc')
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Fitting 5 folds for each of 22 candidates, totalling 110 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[CV] alpha=1e-11 .....................................................
[CV] ............. alpha=1e-11, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ............. alpha=1e-11, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.1s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ............ alpha=1e-11, score=0.9094128647360861, total=   0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.3s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ............ alpha=1e-11, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.4s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ............ alpha=1e-11, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.5s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............. alpha=1e-10, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.6s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............. alpha=1e-10, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.7s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.9094128647360861, total=   0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.8s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.9s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    1.0s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............. alpha=1e-10, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    1.1s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............. alpha=1e-10, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    1.2s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.9094128647360861, total=   0.0s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    1.3s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
```

```
[CV] ........... alpha=1e-10, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    1.4s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ........... alpha=1e-10, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:    1.5s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ............ alpha=1e-09, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed:    1.6s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ............ alpha=1e-09, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed:    1.7s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ........... alpha=1e-09, score=0.9094128647360861, total=   0.0s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    1.8s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ........... alpha=1e-09, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed:    1.9s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ........... alpha=1e-09, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    2.0s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ............ alpha=1e-08, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    2.2s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ............ alpha=1e-08, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed:    2.3s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ........... alpha=1e-08, score=0.9094128647360861, total=   0.0s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:    2.4s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ........... alpha=1e-08, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:    2.5s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ........... alpha=1e-08, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    2.7s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    2.8s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    2.9s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ........... alpha=1e-07, score=0.9094128647360861, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    3.0s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    3.1s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    3.2s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............. alpha=1e-06, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    3.3s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............. alpha=1e-06, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    3.4s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............ alpha=1e-06, score=0.9094128647360861, total=   0.0s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    3.5s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............ alpha=1e-06, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    3.6s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............ alpha=1e-06, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    3.7s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............. alpha=1e-05, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    3.8s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............. alpha=1e-05, score=0.895559643229585, total=   0.0s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:    3.9s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............ alpha=1e-05, score=0.9094128647360861, total=   0.0s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:    4.0s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............ alpha=1e-05, score=0.9029790428364183, total=   0.0s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:    4.1s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............ alpha=1e-05, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    4.2s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ............ alpha=0.0001, score=0.895692105573748, total=   0.0s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:    4.3s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.8955597401296174, total=   0.0s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    4.4s remaining:
```

```
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.9094127678271703, total=   0.0s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:    4.5s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.9029790913164999, total=   0.0s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:    4.7s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.8963891938286019, total=   0.0s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:    4.8s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.8956922993738126, total=   0.0s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:    4.9s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.8955594009795043, total=   0.0s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:    5.0s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.9094126224637968, total=   0.0s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    5.1s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.9029789943563369, total=   0.0s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:    5.3s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.8963894362290099, total=   0.0s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:    5.4s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] .............. alpha=0.01, score=0.895693655974265, total=   0.0s
[Parallel(n_jobs=1)]: Done  51 out of  51 | elapsed:    5.5s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] ............. alpha=0.01, score=0.8955608060299727, total=   0.0s
[Parallel(n_jobs=1)]: Done  52 out of  52 | elapsed:    5.6s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] ............. alpha=0.01, score=0.9094131070083752, total=   0.0s
[Parallel(n_jobs=1)]: Done  53 out of  53 | elapsed:    5.7s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] ............. alpha=0.01, score=0.9029804487587838, total=   0.0s
[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:    5.8s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] ............. alpha=0.01, score=0.8963907451912121, total=   0.0s
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:    5.9s remaining:
0.0s
[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.8958758764850172, total=   0.0s
[Parallel(n_jobs=1)]: Done  56 out of  56 | elapsed:    6.0s remaining:
0.0s
```

```
[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.8956148277979836, total=   0.0s
[Parallel(n_jobs=1)]: Done  57 out of  57 | elapsed:    6.1s remaining:
0.0s
[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.9093528538900402, total=   0.0s
[Parallel(n_jobs=1)]: Done  58 out of  58 | elapsed:    6.2s remaining:
0.0s
[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.9029917446177899, total=   0.0s
[Parallel(n_jobs=1)]: Done  59 out of  59 | elapsed:    6.3s remaining:
0.0s
[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.8965198961285165, total=   0.0s
[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:    6.4s remaining:
0.0s
[CV] alpha=10 ........................................................
[CV] ................ alpha=10, score=0.896906262978548, total=   0.2s
[Parallel(n_jobs=1)]: Done  61 out of  61 | elapsed:    6.8s remaining:
0.0s
[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.8957779105523553, total=   0.0s
[Parallel(n_jobs=1)]: Done  62 out of  62 | elapsed:    6.9s remaining:
0.0s
[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.9084498081591105, total=   0.0s
[Parallel(n_jobs=1)]: Done  63 out of  63 | elapsed:    7.1s remaining:
0.0s
[CV] alpha=10 ........................................................
[CV] ................ alpha=10, score=0.902590765863119, total=   0.0s
[Parallel(n_jobs=1)]: Done  64 out of  64 | elapsed:    7.2s remaining:
0.0s
[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.8969353704275769, total=   0.0s
[Parallel(n_jobs=1)]: Done  65 out of  65 | elapsed:    7.3s remaining:
0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.8834726238997728, total=   0.0s
[Parallel(n_jobs=1)]: Done  66 out of  66 | elapsed:    7.4s remaining:
0.0s
[CV] alpha=100 .......................................................
[CV] ............... alpha=100, score=0.878786296087351, total=   0.0s
[Parallel(n_jobs=1)]: Done  67 out of  67 | elapsed:    7.5s remaining:
0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.8869992134872402, total=   0.0s
[Parallel(n_jobs=1)]: Done  68 out of  68 | elapsed:    7.6s remaining:
0.0s
[CV] alpha=100 .......................................................
[CV] ............... alpha=100, score=0.883762459623364, total=   0.0s
[Parallel(n_jobs=1)]: Done  69 out of  69 | elapsed:    7.7s remaining:
0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.8798669153888772, total=   0.0s
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:    7.8s remaining:
0.0s
[CV] alpha=1000 ......................................................
```

```
[CV] ............. alpha=1000, score=0.6619423621165834, total=   0.0s
[Parallel(n_jobs=1)]: Done  71 out of  71 | elapsed:    7.9s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] ............. alpha=1000, score=0.6722991586945395, total=   0.0s
[Parallel(n_jobs=1)]: Done  72 out of  72 | elapsed:    8.0s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] ............. alpha=1000, score=0.6660588054929525, total=   0.0s
[Parallel(n_jobs=1)]: Done  73 out of  73 | elapsed:    8.1s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] .............. alpha=1000, score=0.669610062099106, total=   0.0s
[Parallel(n_jobs=1)]: Done  74 out of  74 | elapsed:    8.2s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] ............. alpha=1000, score=0.6578182033785962, total=   0.0s
[Parallel(n_jobs=1)]: Done  75 out of  75 | elapsed:    8.3s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.5371599026891116, total=   0.0s
[Parallel(n_jobs=1)]: Done  76 out of  76 | elapsed:    8.4s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.5474771640538865, total=   0.0s
[Parallel(n_jobs=1)]: Done  77 out of  77 | elapsed:    8.5s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.5467382736335649, total=   0.0s
[Parallel(n_jobs=1)]: Done  78 out of  78 | elapsed:    8.6s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.5362745180740531, total=   0.0s
[Parallel(n_jobs=1)]: Done  79 out of  79 | elapsed:    8.7s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.5348505368781193, total=   0.0s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    8.9s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.5103421888980858, total=   0.0s
[Parallel(n_jobs=1)]: Done  81 out of  81 | elapsed:    9.0s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.5201479876673392, total=   0.0s
[Parallel(n_jobs=1)]: Done  82 out of  82 | elapsed:    9.1s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.5204558246522326, total=   0.0s
[Parallel(n_jobs=1)]: Done  83 out of  83 | elapsed:    9.3s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.5081355394485332, total=   0.0s
[Parallel(n_jobs=1)]: Done  84 out of  84 | elapsed:    9.4s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.5088674432400053, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done  85 out of  85 | elapsed:    9.6s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.5060547500686537, total=   0.0s
[Parallel(n_jobs=1)]: Done  86 out of  86 | elapsed:    9.7s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.5157379702970393, total=   0.0s
[Parallel(n_jobs=1)]: Done  87 out of  87 | elapsed:    9.8s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.5162362650993781, total=   0.0s
[Parallel(n_jobs=1)]: Done  88 out of  88 | elapsed:    9.9s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] ........... alpha=1000000, score=0.503689915968505, total=   0.0s
[Parallel(n_jobs=1)]: Done  89 out of  89 | elapsed:   10.0s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.5046957807009328, total=   0.0s
[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:   10.1s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.5055912287641157, total=   0.0s
[Parallel(n_jobs=1)]: Done  91 out of  91 | elapsed:   10.2s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] .......... alpha=10000000, score=0.515254778285943, total=   0.0s
[Parallel(n_jobs=1)]: Done  92 out of  92 | elapsed:   10.3s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.5157701816693296, total=   0.0s
[Parallel(n_jobs=1)]: Done  93 out of  93 | elapsed:   10.4s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.5032051636328801, total=   0.0s
[Parallel(n_jobs=1)]: Done  94 out of  94 | elapsed:   10.5s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.5042417647370236, total=   0.0s
[Parallel(n_jobs=1)]: Done  95 out of  95 | elapsed:   10.6s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.5055416159475747, total=   0.0s
[Parallel(n_jobs=1)]: Done  96 out of  96 | elapsed:   10.7s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.5152058922196445, total=   0.0s
[Parallel(n_jobs=1)]: Done  97 out of  97 | elapsed:   10.8s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.5157247798423253, total=   0.0s
[Parallel(n_jobs=1)]: Done  98 out of  98 | elapsed:   10.9s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.5031557139496781, total=   0.0s
[Parallel(n_jobs=1)]: Done  99 out of  99 | elapsed:   11.1s remaining:
```

```
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.5041961449802657, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.5055360441957171, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.5152011441180615, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.5157212426669023, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.5031513022622551, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.5041917332928427, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.5055355596955556, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.5152008291929564, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.5157207338950949, total=   0.2s
[CV] alpha=10000000000 ...............................................
[CV] ........ alpha=10000000000, score=0.50315052658095, total=   0.1s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.5041909576115376, total=   0.0s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:   12.8s finished
```

In [13]:

```
#print the results of the gridsearchCV
print(return_63[0])  # best score
print(return_63[1])  # best alpha = 10
print(return_63[2])
```

```
0.9001319166904739
{'alpha': 10}
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e),
       fit_params=None, iid='warn', n_jobs=1,
       param_grid={'alpha': [1e-11, 1e-10, 1e-10, 1e-09, 1e-08, 1e-07, 1e-0
6, 1e-05, 0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000, 100000, 1000000, 10
000000, 100000000, 1000000000, 10000000000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=100)
```

In [14]:
```
#calculate roauc score varying alpha
mnbayes.calcrocaucscore_naivebayes(100000000)
```

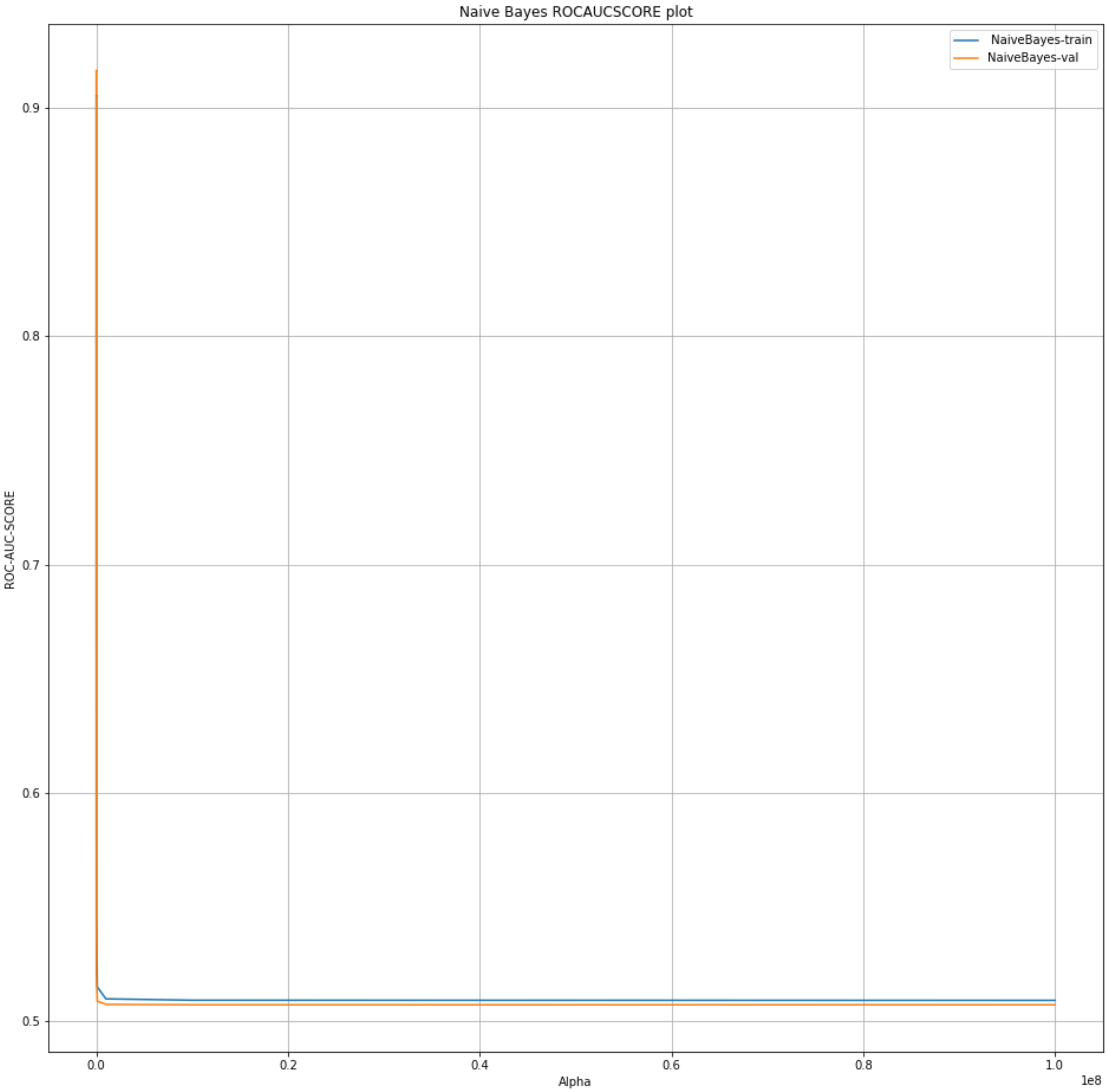
```
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
```

```
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Function exiting...
```

In [15]:
```
print(mnbayes.rocaucscoretrn)
print(mnbayes.rocaucscoreval)
print( mnbayes.NBayes_alpha)
```

```
[0.9053365680876713, 0.9053365680876713, 0.9053365680876713, 0.905336568087
6713, 0.9053365680876713, 0.9053365700261883, 0.9053365758417397, 0.9053365
797187738, 0.90533656227212, 0.9053368045867547, 0.9053378688326301, 0.9053
488893022151, 0.9051831557846834, 0.8919818447950495, 0.693085396430503, 0.
5464037119686502, 0.514971675280183, 0.5096905828988257, 0.509104010893411
5, 0.5090437501515437]
[0.9163071019388954, 0.9163071019388954, 0.9163071019388954, 0.916307101938
8954, 0.9163071019388954, 0.9163071327455263, 0.9163071327455263, 0.9163071
943587884, 0.9163067014526929, 0.916305715640502, 0.9163012948889582, 0.916
1995867968202, 0.9123514072961931, 0.8161240150503948, 0.5781911418366863,
0.5203568985165499, 0.5087069705301304, 0.5072543146535068, 0.5071018218302
195, 0.5070857407688547]
[1e-11, 9.999999999999999e-11, 9.999999999999999e-10, 9.999999999999999e-0
9, 9.999999999999998e-08, 9.999999999999997e-07, 9.999999999999997e-06, 9.9
99999999999998e-05, 0.0009999999999999998, 0.009999999999999998, 0.09999999
999999998, 0.9999999999999998, 9.999999999999998, 99.99999999999999, 999.99
99999999999, 9999.999999999998, 99999.99999999999, 999999.9999999999, 99999
99.999999998, 99999999.99999999]
```

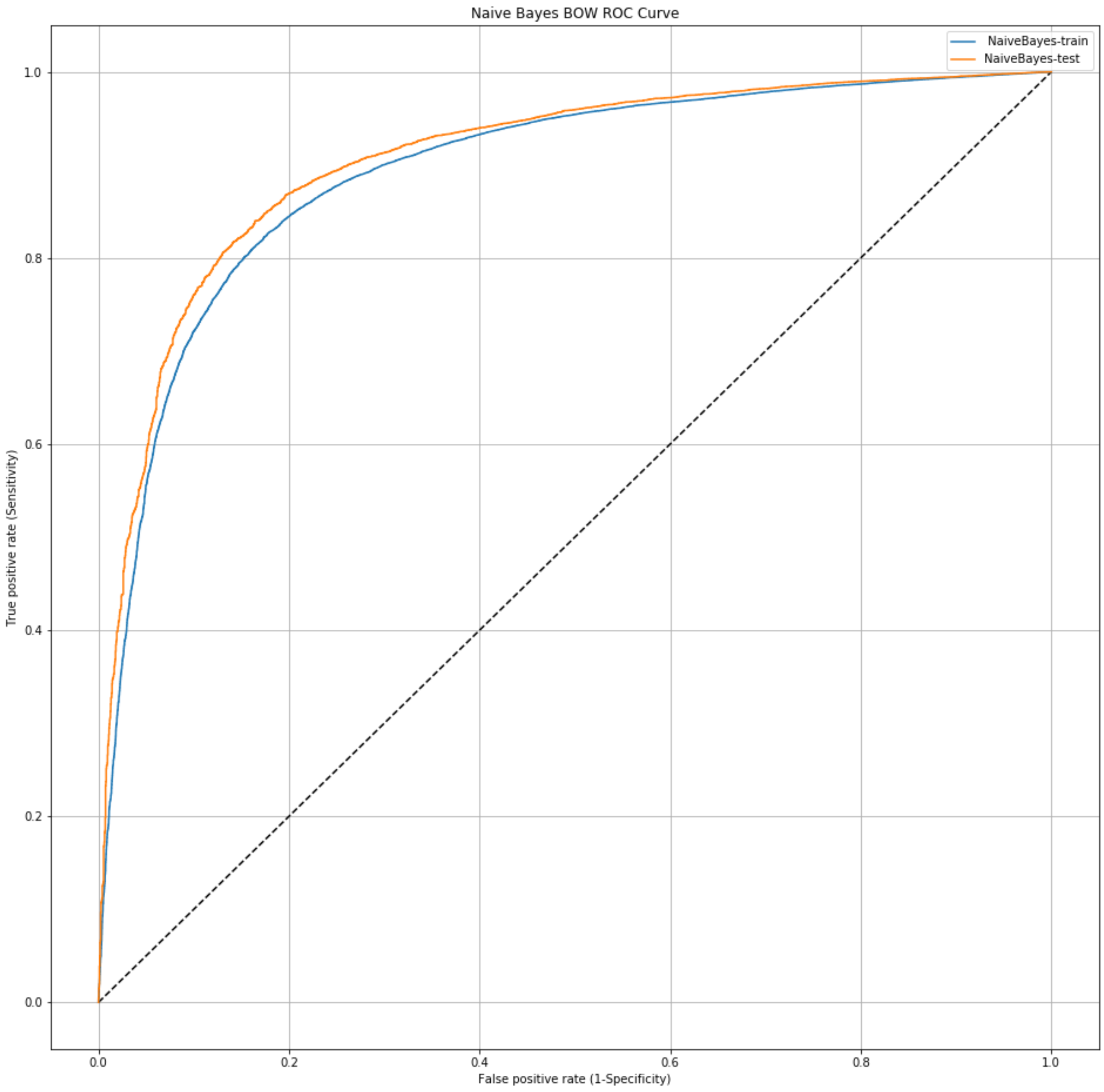
In [16]:
```
# display rocauc graph
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Naive Bayes ROCAUCSCORE plot'
displaygraph.legnd_1 = ' NaiveBayes-train'
displaygraph.legnd_2 = 'NaiveBayes-val'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='Alpha'
displaygraph.label_y='ROC-AUC-SCORE'
displaygraph.Xdata = mnbayes.NBayes_alpha
displaygraph.ydatatrn = mnbayes.rocaucscoretrn
displaygraph.ydataval = mnbayes.rocaucscoreval
displaygraph.rocacuscoregraph()
```



In [21]:
```
#process test data using the hyper parameter tuned alpha value of 10
mnbayes.actualClasifier_naivebayes(10)
```

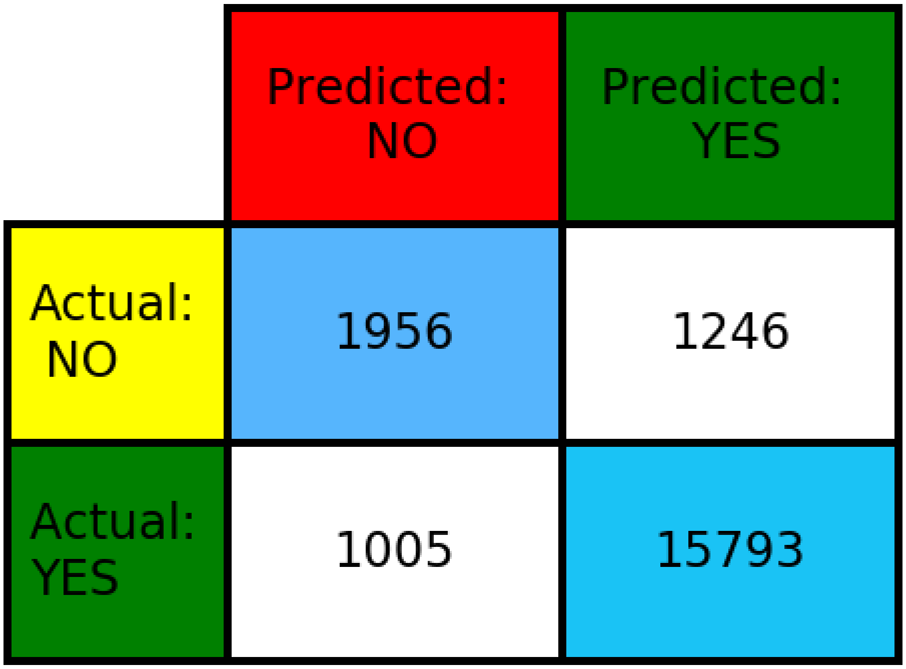
In [22]:

```
# display the roc auc cure
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Naive Bayes BOW ROC Curve'
displaygraph.legnd_1 = ' NaiveBayes-train'
displaygraph.legnd_2 = 'NaiveBayes-test'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='False positive rate (1-Specificity)'
displaygraph.label_y='True positive rate (Sensitivity)'
displaygraph.constructgraph(mnbayes.roc_curve_test['fpr_trn'],mnbayes.roc_c
urve_test['tpr_trn'],\
                            mnbayes.roc_curve_test['fpr'],mnbayes.roc_curve
_test['tpr'])
```



In [23]:

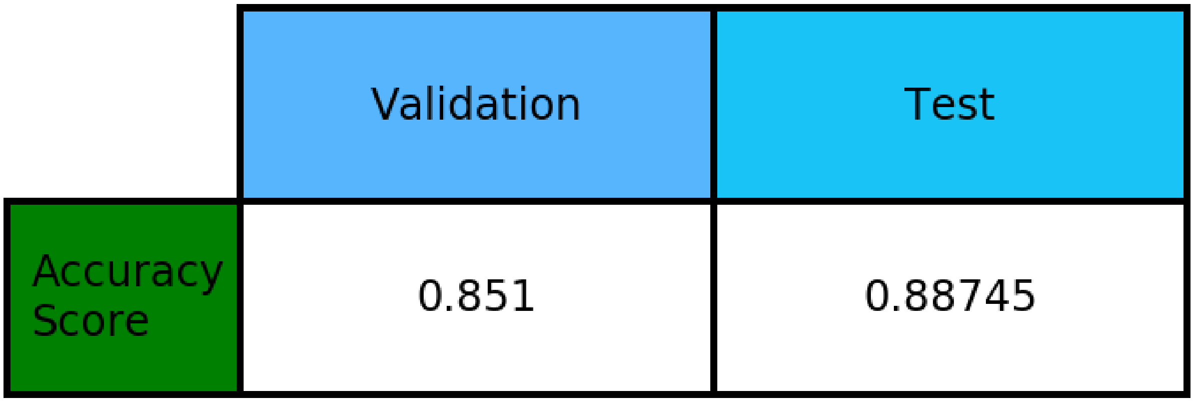
```
#display the confusion matrix for test data
data = [[mnbayes.confsnmtxytstpred['tn'] ,mnbayes.confsnmtxytstpred['fn']],
[mnbayes.confsnmtxytstpred['fp'],mnbayes.confsnmtxytstpred['tp']]]
displaygraph.draw_table(data)
```



	Predicted: NO	Predicted: YES
Actual: NO	1956	1246
Actual: YES	1005	15793

In [24]:

```
#displaying the accuracy score for validation and test data
data1= [[mnbayes.accuracy_score_val,mnbayes.accuracy_score_test]]
displaygraph.draw_accscore(data1)
```



	Validation	Test
Accuracy Score	0.851	0.88745

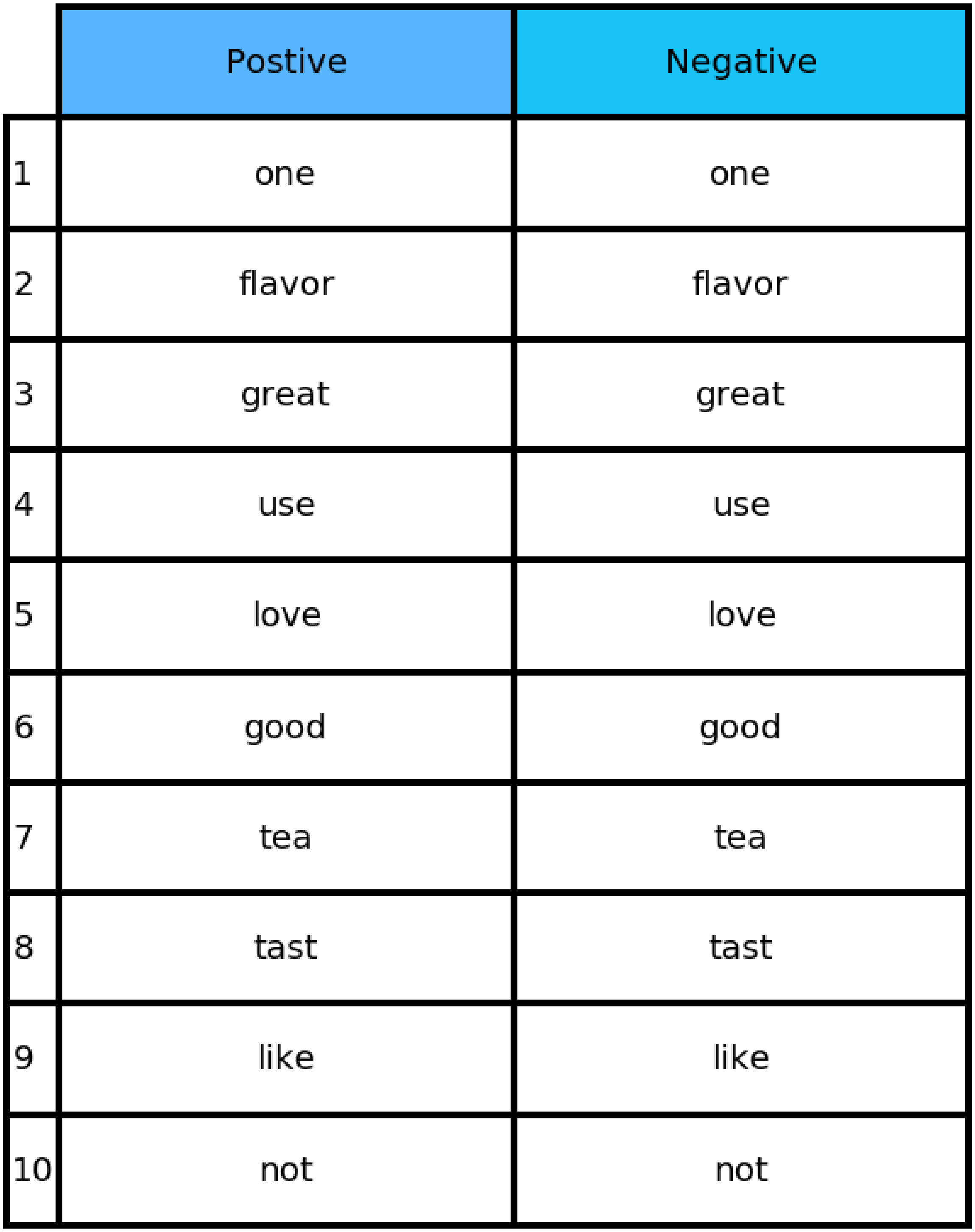
In [63]:

```
mnbayes.outputdir = 'E:/appliedaiacourse/assignments/assign4-MultinomialNBa
yes'
mnbayes.exportopdatatocsv('NBayes_alpha',mnbayes.NBayes_alpha)
mnbayes.exportopdatatocsv('yprdprobatrn',mnbayes.yprdprobatrn)
mnbayes.exportopdatatocsv('yprdprobaval',mnbayes.yprdprobaval)
mnbayes.exportopdatatocsv('ytrn_predprob_actclf',mnbayes.ytrn_predprob_actc
lf)
mnbayes.exportopdatatocsv('ytst_predprob_actclf',mnbayes.ytst_predprob_actc
lf)
mnbayes.exportopdatatocsv('rocaucscoretrn',mnbayes.rocaucscoretrn)
mnbayes.exportopdatatocsv('rocaucscoreval',mnbayes.rocaucscoreval)
mnbayes.exportopdatatocsv('predicted',mnbayes.predicted)
mnbayes.exportopdatatocsv('test_predict',mnbayes.test_predict)
```

[5.1.1] Top 10 important features of positive class from SET 1

In [25]:

```
class_labels = mnbayes_clf.classes_
feature_names = mnbayes.count_vect.get_feature_names()
top10_negve = sorted(zip(mnbayes_clf.coef_[0], feature_names))[-10:]
top10_posve = sorted(zip(mnbayes_clf.coef_[0], feature_names))[-10:]
feat_pos=[]
feat_neg=[]
features=[]
for coef,feat in (top10_negve):
    feat_pos.append(feat)

for cef,feat in (top10_posve):
    feat_neg.append(feat)

i=0
while i< int(len(feat_pos)):
    feat_item=[]
    feat_item.append(feat_pos[i])
    feat_item.append(feat_neg[i])
    features.append(feat_item)
    i +=1

displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.draw_posnegwords(features)
```



	Postive	Negative
1	one	one
2	flavor	flavor
3	great	great
4	use	use
5	love	love
6	good	good
7	tea	tea
8	tast	tast
9	like	like
10	not	not

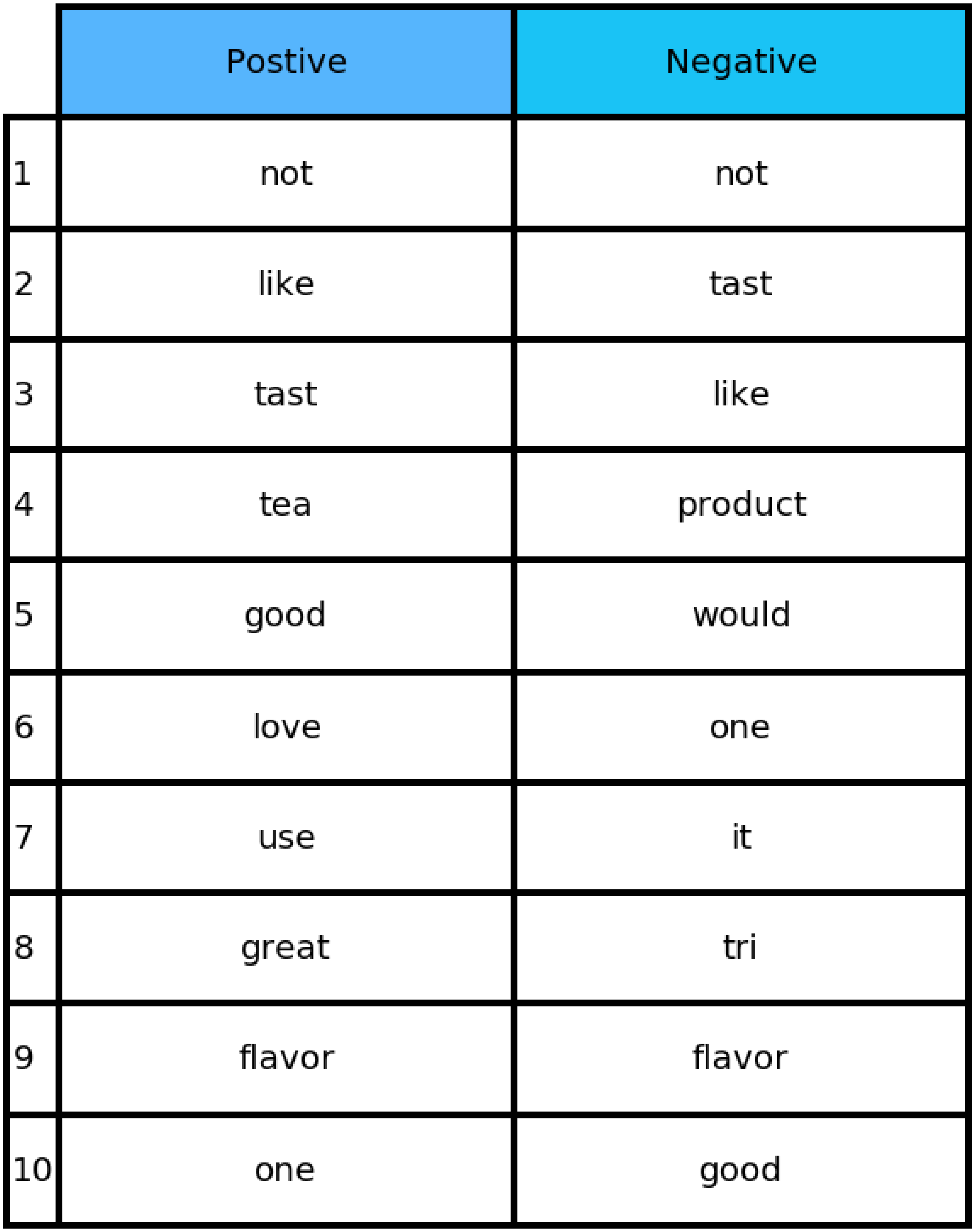
[5.1.2] Top 10 important features of negative class from SET 1

In [0]:

```
# Please write all the code with proper documentation
```

In [27]:

```
feat1_pos=[]
feat0_neg=[]
features1=[]

class_labels = mnbayes_clf.classes_
feature_names = mnbayes.count_vect.get_feature_names()
top10n_neg = sorted(zip(mnbayes_clf.feature_count_[0], feature_names),rever
se=True)[:10]
top10n_pos = sorted(zip(mnbayes_clf.feature_count_[1], feature_names),rever
se=True)[:10]

for coef, feat in top10n_neg:
    feat0_neg.append(feat)


for coef, feat in top10n_pos:
    feat1_pos.append(feat)

i=0
while i< int(len(feat1_pos)):
    feat_item=[]
    feat_item.append(feat1_pos[i])
    feat_item.append(feat0_neg[i])
    features1.append(feat_item)
    i +=1

displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.draw_posnegwords(features1)
```



	Postive	Negative
1	not	not
2	like	tast
3	tast	like
4	tea	product
5	good	would
6	love	one
7	use	it
8	great	tri
9	flavor	flavor
10	one	good

[5.1.3] Feature Engineering SET 1

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool. If doesn’t require any training data but is constructed from a generalizable, valence-based, human-curated gold standard sentiment lexicon.The Compound score is a metric that calculates the sum of all the positive/negative/neutral ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive).I have used the compund score on the Summary text as a new feature . I have also added the len of summary text as the second new feature.

In [33]:

```
import re

score_summ=[]
len_summ=[]
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
for count,summ in final['Summary'].iteritems():
    #print(len(final['Summary'][count]),final['Summary'][count])
    #len_summ = len(re.findall(r'\w+', final['Summary'][count])),final['Sum
mary'][count]
    len_summ.append(len(re.findall(r'\w+', final['Summary'][count])))
    if (analyser.polarity_scores(final['Summary'][count]))['compound'] < 0:
        score_summ.append(0.0)
    else:
        score_summ.append((analyser.polarity_scores(final['Summary'][count
]))['compound'])
```

In [34]:

```
print(final.shape)
print(len(score_summ))
print(len(len_summ))
```

```
(364171, 10)
364171
364171
```

Adding two new featues

In [35]:

```
new_scoresumm = pd.DataFrame((np.asarray(score_summ)).reshape(-1,1))
new_lensumm = pd.DataFrame((np.asarray(len_summ)).reshape(-1,1))
final_fteng = final.append({'score_summ':new_scoresumm}, ignore_index=True)
final_fteng= final_fteng.append({'len_summ':new_lensumm}, ignore_index=True
)
```

In [38]:

```
# cross checking the final dimensions
print(final_fteng.columns)
print(final_fteng.shape)
```

```
Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
       'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text',
       'score_summ', 'len_summ'],
      dtype='object')
(364173, 12)
```

In [39]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# <br /><br /> ==> after the above steps, we are getting "br br"
# we are including them into stop words list
# instead of <br /> if we have <br/> these tags would have revmoved in the
 1st step

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours'
, 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have'
, 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'be
cause', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'int
o', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on'
, 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so',
'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "sho
uld've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'did
n', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "sh
ouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

In [40]:

```
# get 100k rows for processig
data_set = final_fteng[:100000]
f10Xdata=data_set['Text']
f10ydata=data_set['Score']
```

In [41]:

```
print(type(data_set))
print(data_set.shape)
print(type(f10Xdata))
print(len(f10ydata))
```

```
<class 'pandas.core.frame.DataFrame'>
(100000, 12)
<class 'pandas.core.series.Series'>
100000
```

In [42]:

```
#function to split the data into train/test/cross validate
def splitdatasets(xdata,ydata):
    #split into train and test sets (80/20)
    X_data = xdata
    y_data = ydata

    X_train8, X_test8, y_train8, y_test8 = train_test_split(X_data, y_data,
stratify=y_data,test_size=0.2, random_state=42)
    X_trn8, X_val8, y_trn8, y_val8   = train_test_split(X_train8, y_train8,
test_size=0.2, random_state=42)


    return [X_trn8,X_test8,X_val8,y_trn8, y_test8,y_val8]
```

In [43]:

```
# calling functon to split data into train , test and validation
return_list14 = splitdatasets(f10Xdata,f10ydata)
xtrain = return_list14[0]
xtest  = return_list14[1]
xval  = return_list14[2]
y_trn = return_list14[3]
y_test = return_list14[4]
y_val = return_list14[5]
```

In [44]:

```
"""
    Set of functions to pre-proces text
"""
dictexpand = {row[0] : row[1] for _, row in pd.read_csv("E:/appliedaiacours
e/assignments/dblite/expansions2.txt").iterrows()}
contractions_re = re.compile('(%s)' % '|'.join(dictexpand.keys()))


# remove html
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

# remove any text between square brackers
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    return text

 # this expands contractions eg y'a'' to you all
def expand_contractions(s, contractions_dict=dictexpand):
    #print(s)
    def replace(match):
        #print(match)
        return contractions_dict[match.group(0)]
    #print(contractions_re.sub(replace, s)+ "|\t" + s)
    return contractions_re.sub(replace, s)

# remove common words
def remove_cmn_words(sumtext) :
    text = ' '.join(e.lower() for e in sumtext.split() if e.lower() not in
stopwords)
    return text

def remove_num_spechar(text):
    #remove words with numbers
    text = re.sub('[^A-Za-z0-9]+', ' ', text)
    #remove spacial character:
    text = re.sub("\S*\d\S*", "", text).strip()
    return text

# tokenizing and porter stemming
def stem_word(sumtext):
    # this instantiates the Porter stemmer
    #porter=PorterStemmer()
    stemer= SnowballStemmer("english")

    sumtxt_token = word_tokenize(sumtext)
    smtxt_stemed =[]
    for token in sumtxt_token:
        #smtxt_stemed.append(porter.stem(token))
        smtxt_stemed.append(stemer.stem(token))
        smtxt_stemed.append(" ")
    return "".join(smtxt_stemed)

#print(contractions_re)


"""
main loop that processes text in summary and stores in sumpreproc
at the end we print for a simple check

Expansions2.txt is the file that contains a dictionary of contractions and
expansions. This  approach allows us to add another items later on.
"""

def preproces_txt(textcorpus):

    sumwordcnt = []
    sumscore = []
    sumpreproc = []
    sumpreproc2 = []
    unstemmed_summ = []
    i=0

    all_positive_words=[] # store words from +ve reviews
    all_negative_words=[] # store words from -ve reviews

    for sumstr in textcorpus:  #final['Text'].values :
        prcdtxt1 = denoise_text(sumstr)
        prcdtxt2 = remove_cmn_words(prcdtxt1)
        prcdtxt3 = expand_contractions(prcdtxt2)
        prcdtxt4 = remove_num_spechar(prcdtxt3)
        # unstemmed_stem will contain the summary text before stemming
        # used for bi-grams and n-grams
        unstemmed_summ.append(prcdtxt4)

        prcdtxt4 = stem_word(prcdtxt4)

        sumpreproc.append(prcdtxt4)

        sumwordcnt.append(len(sumstr.split(" ")))
        if (final['Score'].values)[i] == 1:
            all_positive_words.append(prcdtxt4) #list of all words used to
 describe positive reviews
        if(final['Score'].values)[i] == 0:
            all_negative_words.append(prcdtxt4) #list of all words used to
 describe negative reviews reviews
        i = i + 1
        print("Processed {0} word\n".format(i))
    print(len(sumpreproc))
    print("Process Complete...")
    return sumpreproc
```

In [70]:

```
print(xtrain[:10])
```

```
516646    I have ordered the liquid coffee creamers in t...
295864    I made this spaghetti for my family when my ki...
372255    So happy to have found these low carb snack pa...
197368    Out of all their different kinds of cookies, t...
371961    These are a great quick meal, buying in bulk i...
476358    It taste a little less like sugar and more lik...
438301    This tea is just so so so so so so so gooodddd...
131253    This is a lovely pasta made with a different w...
195814    I ordered this to use in my popcorn machine.  ...
189039    Premium chocolates (such as Black Panther) hav...
Name: Text, dtype: object
```

preprocess the data corpus stored in train/test/cross-validate

X_train = preproces_txt(xtrain) X_test = preproces_txt(xtest) X_val = preproces_txt(xval)

In [46]:

```
# function to write data to external storage for easier retrieval in subseq
uent runs
def write_data(fnme,opdata):
    fname = 'E:/appliedaiacourse/assignments/dblite/' + fnme
    with open(fname, 'wb') as fp:
        pickle.dump(opdata, fp)
```

In [47]:

```
# actual writing to eternal storage
write_data('pproc_fteng_xtrain',X_train)
write_data('pproc_fteng_xtest',X_test)
write_data('pproc_fteng_xval',X_val)
write_data('fteng_ytrain',y_trn)
write_data('fteng_ytest',y_test)
write_data('fteng_yval',y_val)
```

In [48]:

```
#create the Mutlinomial NaiveBayes object
mnbayes_feateng = multiNaiveBayes()

#create the actual Multinomial NaiveBayes lassifier
mnbayes_feateng_clf = mnbayes_feateng.multNBClasify()
```

In [49]:

```
#set the train/test/validate of the feature engineered data corpus
mnbayes_feateng.xtrain = X_train
mnbayes_feateng.xtest = X_test
mnbayes_feateng.xval = X_val
mnbayes_feateng.ytrain = y_trn
mnbayes_feateng.ytest = y_test
mnbayes_feateng.yval = y_val
```

```
In [50]: #using the Bag of word vectorizer on the data
         mnbayes_feateng.BOWVectorizer()
```

```
some feature names  ['abl', 'absolut', 'acid', 'across', 'actual', 'ad', 'a
dd', 'addict', 'addit', 'advertis']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (64000, 1000)
the number of unique words  1000
```

```
In [51]: # print shapes of input datasets for confirmation
         print((mnbayes_feateng.xtrain).shape)
         print((mnbayes_feateng.xtest).shape)
         print((mnbayes_feateng.xval).shape)
         print((mnbayes_feateng.ytrain).shape)
         print((mnbayes_feateng.ytest).shape)
         print((mnbayes_feateng.yval).shape)
```

```
(64000, 1000)
(20000, 1000)
(16000, 1000)
(64000,)
(20000,)
(16000,)
```

In [52]:

```
# calculate ROCAUCSCORE
mnbayes_feateng.calcrocaucscore_naivebayes(100000000)
```

```
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
```

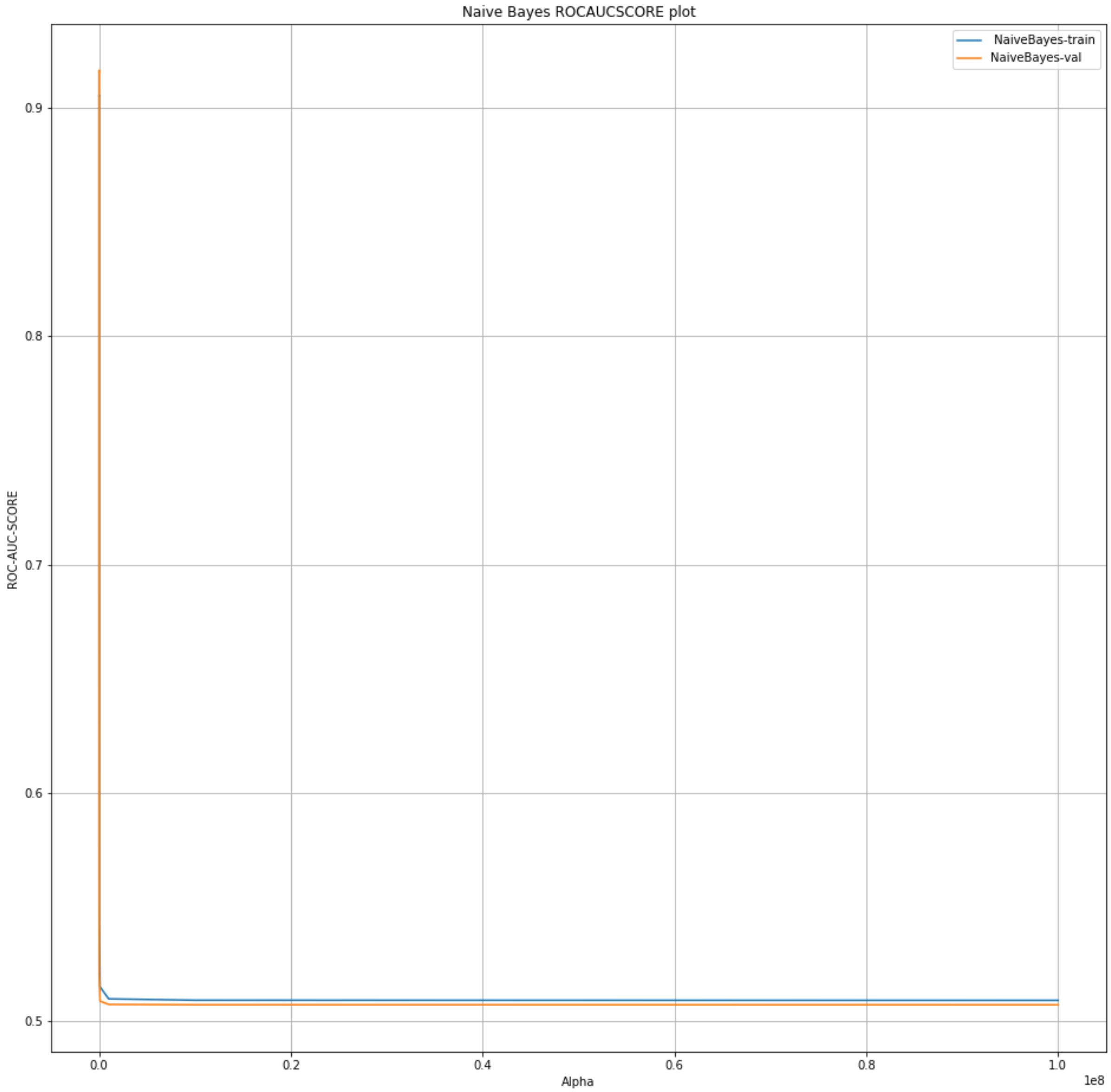
```
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Function exiting...
```

```
In [53]: #print rocaucscore resulst
         print(mnbayes_feateng.rocaucscoretrn)
         print(mnbayes_feateng.rocaucscoreval)
         print( mnbayes_feateng.NBayes_alpha)
```

```
[0.9053365680876713, 0.9053365680876713, 0.9053365680876713, 0.905336568087
6713, 0.9053365680876713, 0.9053365700261883, 0.9053365758417397, 0.9053365
797187738, 0.90533656227212, 0.9053368045867547, 0.9053378688326301, 0.9053
488893022151, 0.9051831557846834, 0.8919818447950495, 0.693085396430503, 0.
5464037119686502, 0.514971675280183, 0.5096905828988257, 0.509104010893411
5, 0.5090437501515437]
[0.9163071019388954, 0.9163071019388954, 0.9163071019388954, 0.916307101938
8954, 0.9163071019388954, 0.9163071327455263, 0.9163071327455263, 0.9163071
943587884, 0.9163067014526929, 0.916305715640502, 0.9163012948889582, 0.916
1995867968202, 0.9123514072961931, 0.8161240150503948, 0.5781911418366863,
0.5203568985165499, 0.5087069705301304, 0.5072543146535068, 0.5071018218302
195, 0.5070857407688547]
[1e-11, 9.999999999999999e-11, 9.999999999999999e-10, 9.999999999999999e-0
9, 9.999999999999998e-08, 9.999999999999997e-07, 9.999999999999997e-06, 9.9
99999999999998e-05, 0.0009999999999999998, 0.009999999999999998, 0.09999999
999999998, 0.9999999999999998, 9.999999999999998, 99.99999999999999, 999.99
99999999999, 9999.999999999998, 99999.99999999999, 999999.9999999999, 99999
99.999999998, 99999999.99999999]
```

In [54]:

```
#display roc auc graph
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Naive Bayes ROCAUCSCORE plot'
displaygraph.legnd_1 = ' NaiveBayes-train'
displaygraph.legnd_2 = 'NaiveBayes-val'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='Alpha'
displaygraph.label_y='ROC-AUC-SCORE'
displaygraph.Xdata = mnbayes_feateng.NBayes_alpha
displaygraph.ydatatrn = mnbayes_feateng.rocaucscoretrn
displaygraph.ydataval = mnbayes_feateng.rocaucscoreval
displaygraph.rocacuscoregraph()
```

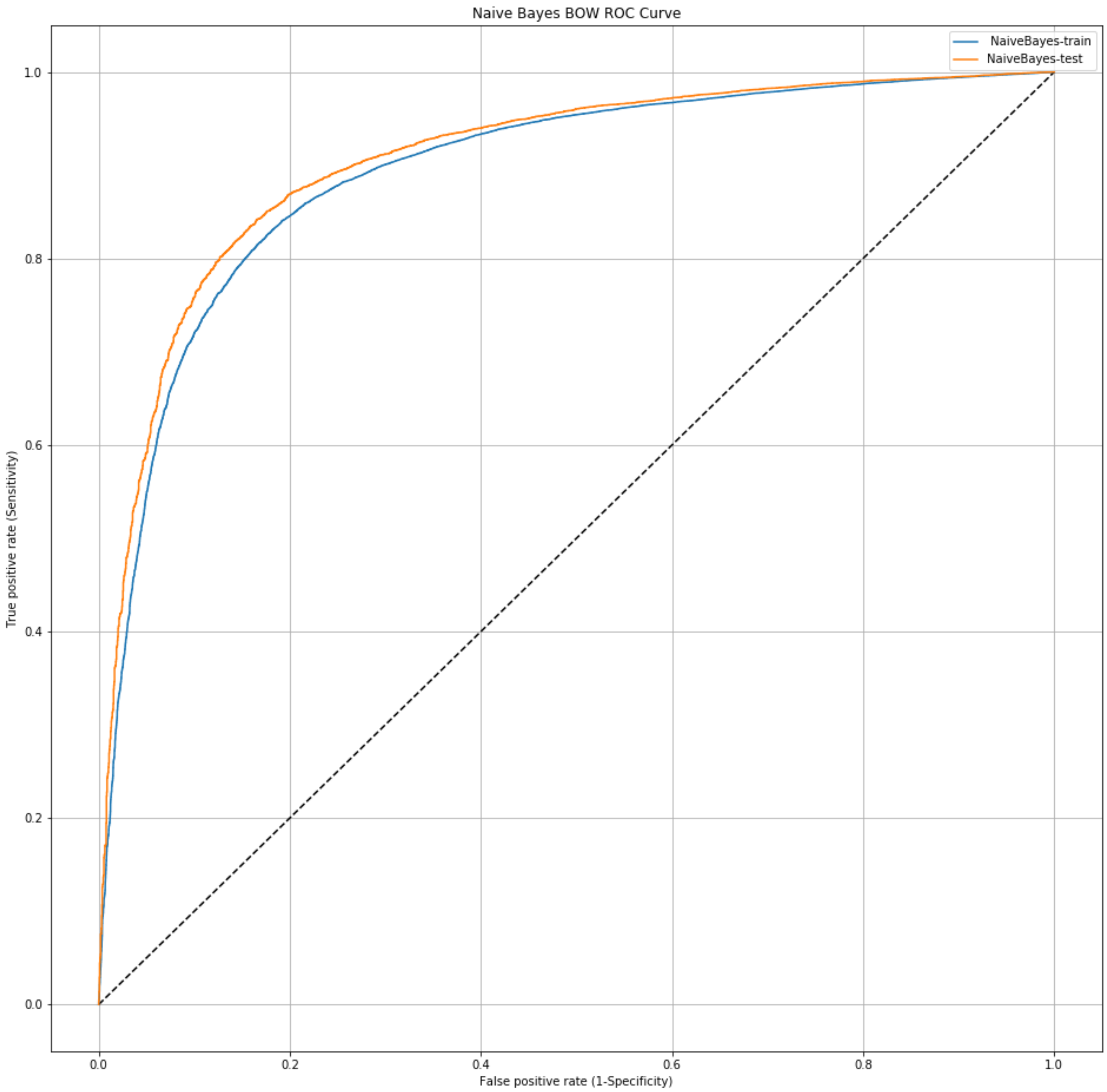


In [55]:

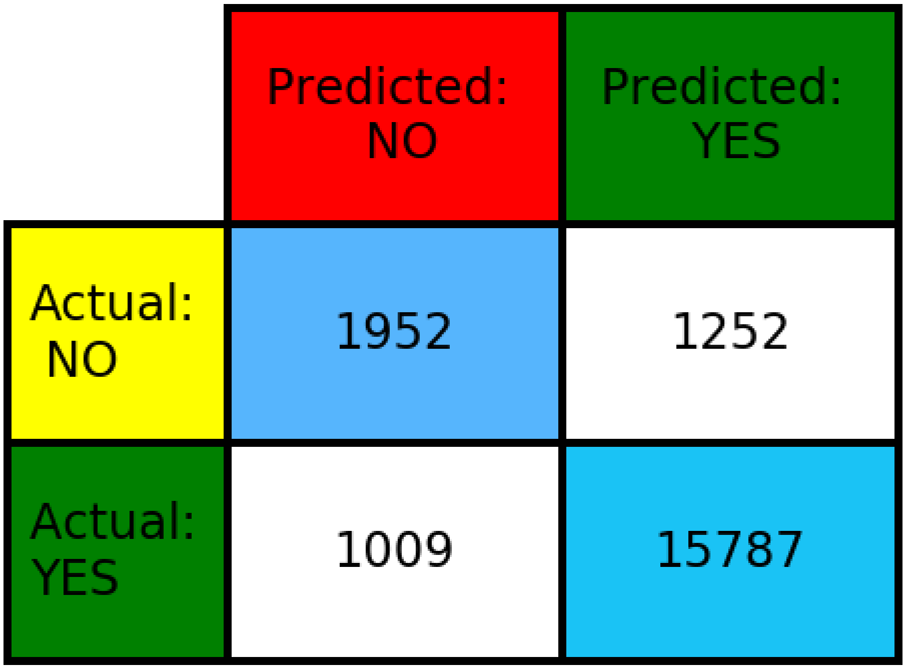
```
#clssify test data with actualClasifier_naivebayes function and hyper param
eter 10
mnbayes_feateng.actualClasifier_naivebayes(10)
```

In [56]:

```
# displayig ROCAUC graphs for test data
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Naive Bayes BOW ROC Curve'
displaygraph.legnd_1 = ' NaiveBayes-train'
displaygraph.legnd_2 = 'NaiveBayes-test'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='False positive rate (1-Specificity)'
displaygraph.label_y='True positive rate (Sensitivity)'
displaygraph.constructgraph(mnbayes_feateng.roc_curve_test['fpr_trn'],mnbay
es_feateng.roc_curve_test['tpr_trn'],\
                            mnbayes_feateng.roc_curve_test['fpr'],mnbayes_f
eateng.roc_curve_test['tpr'])
```

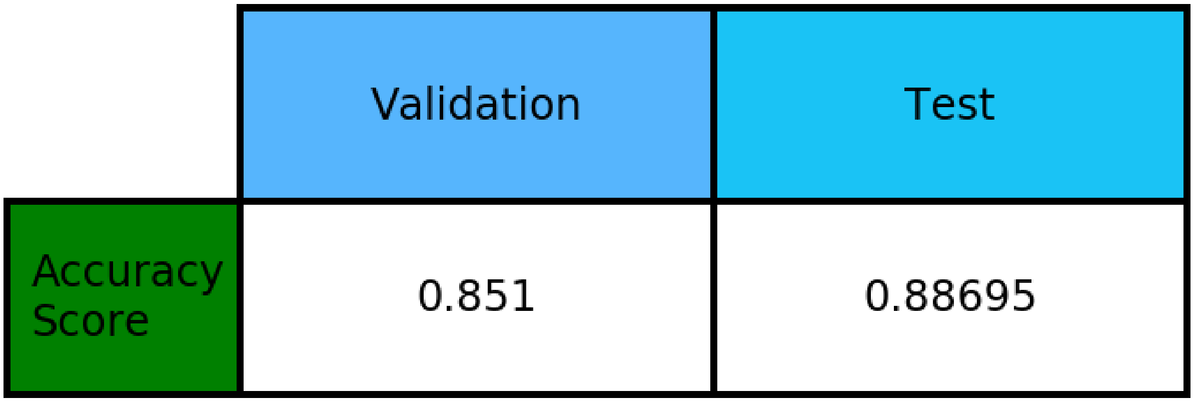


In [57]:
```
data = [[mnbayes_feateng.confsnmtxytstpred['tn'] ,mnbayes_feateng.confsnmtxytstpred['fn']],[mnbayes_feateng.confsnmtxytstpred['fp'],mnbayes_feateng.confsnmtxytstpred['tp']]]

# display confusion matrix for test data
displaygraph.draw_table(data)
```



	Predicted: NO	Predicted: YES
Actual: NO	1952	1252
Actual: YES	1009	15787

In [58]:
```
#display the accuracy score for validation and test data
data1= [[mnbayes_feateng.accuracy_score_val,mnbayes_feateng.accuracy_score_test]]

displaygraph.draw_accscore(data1)
```



	Validation	Test
Accuracy Score	0.851	0.88695

[5.2] Applying Naive Bayes on TFIDF, SET 2

In [0]:
```
# Please write all the code with proper documentation
```

In [59]:

```
#"""
#crate python object for Multinomial Naive Bayes
mnbayes_tfidf = multiNaiveBayes()

#clasifier for the actual classification
mnbayes_tfidf_clf = mnbayes_tfidf.multNBClasify()

#load data
mnbayes_tfidf.xtrain,mnbayes_tfidf.xtest,mnbayes_tfidf.xval, mnbayes_tfidf.
ytrain,mnbayes_tfidf.ytest,mnbayes_tfidf.yval = load_data()

#tfidf vectorizer for the data corpus
mnbayes_tfidf.tfIdfVectorizer()

#print the shapes for confirmation
print((mnbayes_tfidf.xtrain).shape)
print((mnbayes_tfidf.xtest).shape)
print((mnbayes_tfidf.xval).shape)
print((mnbayes_tfidf.ytrain).shape)
print((mnbayes_tfidf.ytest).shape)
print((mnbayes_tfidf.yval).shape)
#print the classifier
print(mnbayes_tfidf.getNBClassifier())

#hyper parameter tune alpha for the tfidf vectorized corpus
return_63 = mnbayes_tfidf.hyperparamtuning([0.00000000001,0.0000000001,0.00
00000001,0.000000001,0.00000001,0.0000001,0.000001,0.00001,0.0001,0.001,0.0
1,1,10,100,1000,10000,100000,1000000,10000000,100000000,1000000000,10000000
000],'roc_auc')

#print the results of the hyper parameter tuning corpus
print(return_63[0])
print(return_63[1])
print(return_63[2])

#calculate roc auc score varying alpha
mnbayes_tfidf.calcrocaucscore_naivebayes(100000000)

#print output data for confirmation
print(mnbayes_tfidf.rocaucscoretrn)
print(mnbayes_tfidf.rocaucscoreval)
print( mnbayes_tfidf.NBayes_alpha)
#"""

# display the rocauc score
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Naive Bayes ROCAUCSCORE plot'
displaygraph.legnd_1 = ' NaiveBayes-train'
displaygraph.legnd_2 = 'NaiveBayes-val'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='Alpha'
displaygraph.label_y='ROC-AUC-SCORE'
displaygraph.Xdata = mnbayes_tfidf.NBayes_alpha
displaygraph.ydatatrn = mnbayes_tfidf.rocaucscoretrn
displaygraph.ydataval = mnbayes_tfidf.rocaucscoreval
displaygraph.rocacuscoregraph()

#process the unseen or test data with the hyper parameter tuned value for a
lpha
mnbayes_tfidf.actualClasifier_naivebayes(1)


# testing code for displayig graphs
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Naive Bayes BOW ROC Curve'
displaygraph.legnd_1 = ' NaiveBayes-train'
displaygraph.legnd_2 = 'NaiveBayes-test'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='False positive rate (1-Specificity)'
displaygraph.label_y='True positive rate (Sensitivity)'
displaygraph.constructgraph(mnbayes_tfidf.roc_curve_test['fpr_trn'],mnbayes
_tfidf.roc_curve_test['tpr_trn'],\
                            mnbayes_tfidf.roc_curve_test['fpr'],mnbayes_tfi
df.roc_curve_test['tpr'])

#display conusion matrix
data = [[mnbayes_tfidf.confsnmtxytstpred['tn'] ,mnbayes_tfidf.confsnmtxytst
pred['fn']],[mnbayes_tfidf.confsnmtxytstpred['fp'],mnbayes_tfidf.confsnmtxy
tstpred['tp']]]
displaygraph.draw_table(data)

#display accuracy score
data1= [[mnbayes_tfidf.accuracy_score_val,mnbayes_tfidf.accuracy_score_test
]]

displaygraph.draw_accscore(data1)
```

```
some sample features(unique words in the corpus) ['ab', 'abandon', 'abc',
'abdomin', 'abil', 'abl', 'abl buy', 'abl chew', 'abl drink', 'abl eat']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text TFIDF vectorizer  (64000, 40440)
the number of unique words including both unigrams and bigrams  40440
(64000, 40440)
(20000, 40440)
(16000, 40440)
(64000,)
(20000,)
(16000,)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Fitting 5 folds for each of 22 candidates, totalling 110 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[CV] alpha=1e-11 .....................................................
[CV] ........... alpha=1e-11, score=0.8341024208147471, total=   0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ........... alpha=1e-11, score=0.8523605187019969, total=   0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.2s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ........... alpha=1e-11, score=0.8484808509920759, total=   0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.3s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ........... alpha=1e-11, score=0.8414333853621568, total=   0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.4s remaining:
0.0s
[CV] alpha=1e-11 .....................................................
[CV] ........... alpha=1e-11, score=0.8372397855861736, total=   0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.6s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ........... alpha=1e-10, score=0.8341024208147471, total=   0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.7s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ........... alpha=1e-10, score=0.8523605187019969, total=   0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.9s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ........... alpha=1e-10, score=0.8484808509920759, total=   0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    1.0s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ........... alpha=1e-10, score=0.8414333853621568, total=   0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.2s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ........... alpha=1e-10, score=0.8372397855861736, total=   0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    1.3s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
```

```
[CV] ............ alpha=1e-10, score=0.8341024208147471, total=   0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    1.4s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.8523605187019969, total=   0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    1.6s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.8484808509920759, total=   0.0s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    1.7s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.8414333853621568, total=   0.0s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    1.9s remaining:
0.0s
[CV] alpha=1e-10 .....................................................
[CV] ............ alpha=1e-10, score=0.8372397855861736, total=   0.0s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:    2.0s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ............ alpha=1e-09, score=0.8409455010962301, total=   0.0s
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed:    2.1s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ............. alpha=1e-09, score=0.858750857807536, total=   0.0s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed:    2.3s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ............ alpha=1e-09, score=0.8552354508706683, total=   0.0s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    2.4s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ............ alpha=1e-09, score=0.8483708850349163, total=   0.0s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed:    2.6s remaining:
0.0s
[CV] alpha=1e-09 .....................................................
[CV] ............ alpha=1e-09, score=0.8441433734418745, total=   0.0s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    2.7s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ............ alpha=1e-08, score=0.8490731124309757, total=   0.0s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    2.8s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ............ alpha=1e-08, score=0.8663234015322221, total=   0.0s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed:    3.0s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ............ alpha=1e-08, score=0.8631616553128181, total=   0.0s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:    3.1s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ............ alpha=1e-08, score=0.8564962921464013, total=   0.0s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:    3.3s remaining:
0.0s
[CV] alpha=1e-08 .....................................................
[CV] ............ alpha=1e-08, score=0.8523692555224145, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    3.4s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.8587636970618167, total=   0.0s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    3.6s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.8753172264807634, total=   0.0s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    3.7s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............. alpha=1e-07, score=0.872410448409182, total=   0.0s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    3.9s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.8660331968328157, total=   0.0s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    4.0s remaining:
0.0s
[CV] alpha=1e-07 .....................................................
[CV] ............ alpha=1e-07, score=0.8621350589139647, total=   0.0s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    4.1s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............ alpha=1e-06, score=0.8702859345533306, total=   0.0s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    4.3s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............ alpha=1e-06, score=0.8859413944728609, total=   0.0s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    4.4s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............ alpha=1e-06, score=0.8832273731249094, total=   0.0s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    4.6s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............. alpha=1e-06, score=0.877249354584674, total=   0.0s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    4.7s remaining:
0.0s
[CV] alpha=1e-06 .....................................................
[CV] ............ alpha=1e-06, score=0.8736130091063046, total=   0.0s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    4.8s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............ alpha=1e-05, score=0.8839441879069923, total=   0.0s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    5.0s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............. alpha=1e-05, score=0.898310392896685, total=   0.0s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:    5.1s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............ alpha=1e-05, score=0.8959378980533712, total=   0.0s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:    5.3s remaining:
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............ alpha=1e-05, score=0.8902595177550542, total=   0.0s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:    5.4s remaining:
```

```
0.0s
[CV] alpha=1e-05 .....................................................
[CV] ............ alpha=1e-05, score=0.8869705805412199, total=   0.0s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    5.5s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.8996857435052268, total=   0.0s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:    5.7s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.9121050330458179, total=   0.0s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    5.8s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.9105057308056387, total=   0.0s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:    5.9s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.9051072214572065, total=   0.0s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:    6.1s remaining:
0.0s
[CV] alpha=0.0001 ....................................................
[CV] ........... alpha=0.0001, score=0.9024211340577033, total=   0.0s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:    6.2s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.9167932019588535, total=   0.0s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:    6.4s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.9265920239257808, total=   0.0s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:    6.5s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.9264420143564743, total=   0.0s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    6.6s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............ alpha=0.001, score=0.9213574539191977, total=   0.0s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:    6.8s remaining:
0.0s
[CV] alpha=0.001 .....................................................
[CV] ............. alpha=0.001, score=0.919451653432574, total=   0.0s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:    6.9s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] ............. alpha=0.01, score=0.9330760895875683, total=   0.0s
[Parallel(n_jobs=1)]: Done  51 out of  51 | elapsed:    7.1s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] ............. alpha=0.01, score=0.9396592839824054, total=   0.0s
[Parallel(n_jobs=1)]: Done  52 out of  52 | elapsed:    7.2s remaining:
0.0s
[CV] alpha=0.01 ......................................................
[CV] ............. alpha=0.01, score=0.9417708728237165, total=   0.0s
[Parallel(n_jobs=1)]: Done  53 out of  53 | elapsed:    7.3s remaining:
0.0s
```

```
[CV] alpha=0.01 .....................................................
[CV] ............. alpha=0.01, score=0.936979529964278, total=   0.0s
[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:    7.5s remaining:
0.0s
[CV] alpha=0.01 .....................................................
[CV] ............ alpha=0.01, score=0.9357624859965284, total=   0.0s
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:    7.6s remaining:
0.0s
[CV] alpha=1 ........................................................
[CV] ............... alpha=1, score=0.9349835667235211, total=   0.0s
[Parallel(n_jobs=1)]: Done  56 out of  56 | elapsed:    7.8s remaining:
0.0s
[CV] alpha=1 ........................................................
[CV] ............... alpha=1, score=0.9411851203411192, total=   0.0s
[Parallel(n_jobs=1)]: Done  57 out of  57 | elapsed:    7.9s remaining:
0.0s
[CV] alpha=1 ........................................................
[CV] ............... alpha=1, score=0.9482351820453364, total=   0.0s
[Parallel(n_jobs=1)]: Done  58 out of  58 | elapsed:    8.1s remaining:
0.0s
[CV] alpha=1 ........................................................
[CV] ............... alpha=1, score=0.9416865580441351, total=   0.0s
[Parallel(n_jobs=1)]: Done  59 out of  59 | elapsed:    8.2s remaining:
0.0s
[CV] alpha=1 ........................................................
[CV] ............... alpha=1, score=0.9399984331237636, total=   0.0s
[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:    8.4s remaining:
0.0s
[CV] alpha=10 .......................................................
[CV] .............. alpha=10, score=0.7843032412867006, total=   0.0s
[Parallel(n_jobs=1)]: Done  61 out of  61 | elapsed:    8.5s remaining:
0.0s
[CV] alpha=10 .......................................................
[CV] .............. alpha=10, score=0.8029018943374917, total=   0.0s
[Parallel(n_jobs=1)]: Done  62 out of  62 | elapsed:    8.6s remaining:
0.0s
[CV] alpha=10 .......................................................
[CV] .............. alpha=10, score=0.8067329988812835, total=   0.0s
[Parallel(n_jobs=1)]: Done  63 out of  63 | elapsed:    8.8s remaining:
0.0s
[CV] alpha=10 .......................................................
[CV] .............. alpha=10, score=0.7975181882722029, total=   0.0s
[Parallel(n_jobs=1)]: Done  64 out of  64 | elapsed:    8.9s remaining:
0.0s
[CV] alpha=10 .......................................................
[CV] .............. alpha=10, score=0.7895332764371095, total=   0.0s
[Parallel(n_jobs=1)]: Done  65 out of  65 | elapsed:    9.1s remaining:
0.0s
[CV] alpha=100 ......................................................
[CV] .............. alpha=100, score=0.666168584350606, total=   0.0s
[Parallel(n_jobs=1)]: Done  66 out of  66 | elapsed:    9.2s remaining:
0.0s
[CV] alpha=100 ......................................................
[CV] ............. alpha=100, score=0.6831378149081475, total=   0.0s
[Parallel(n_jobs=1)]: Done  67 out of  67 | elapsed:    9.4s remaining:
0.0s
[CV] alpha=100 ......................................................
```

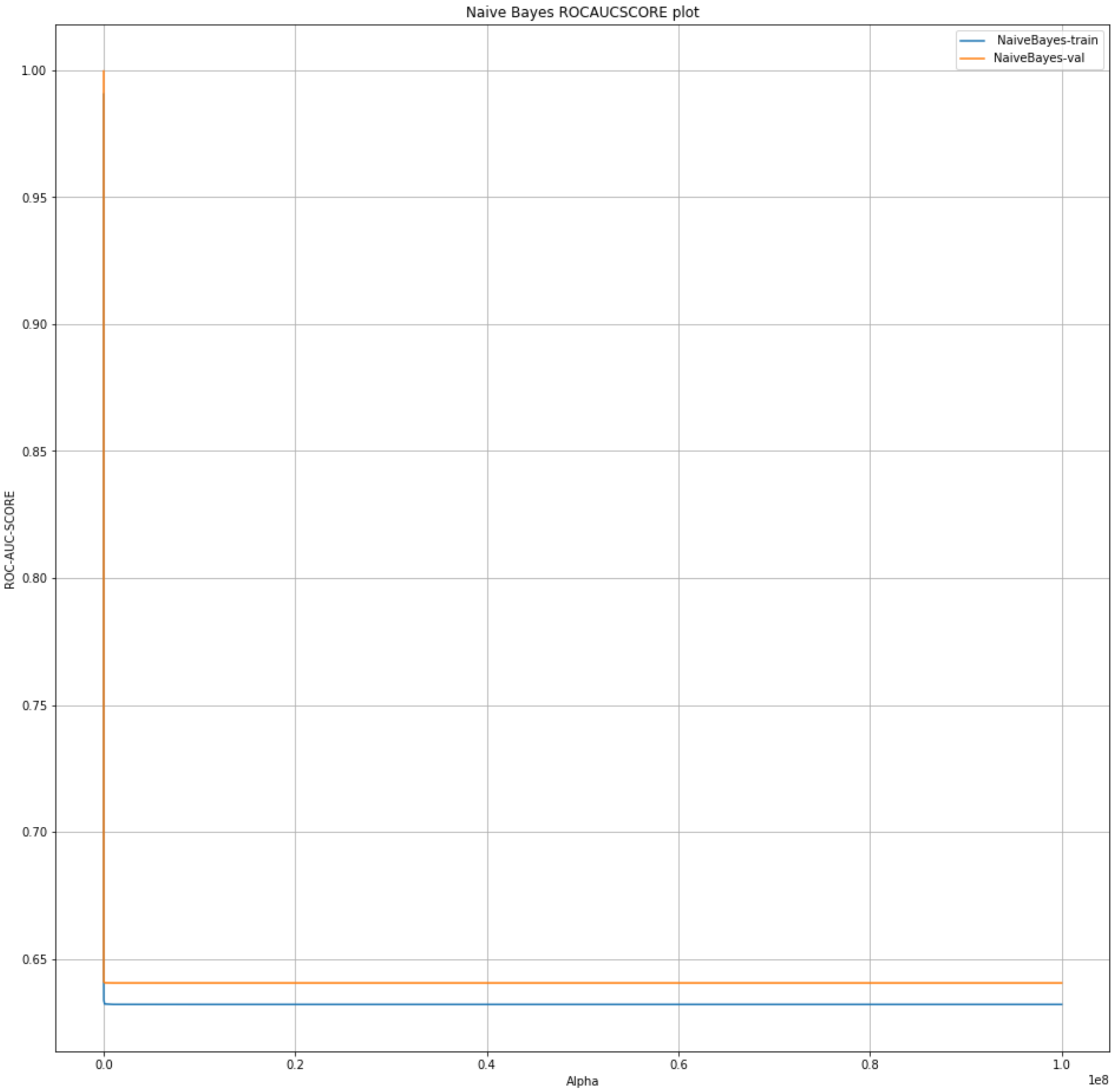
```
[CV] .............. alpha=100, score=0.6856976879858528, total=   0.0s
[Parallel(n_jobs=1)]: Done  68 out of  68 | elapsed:    9.5s remaining:
0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.6709017896325151, total=   0.0s
[Parallel(n_jobs=1)]: Done  69 out of  69 | elapsed:    9.6s remaining:
0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.6665133241686587, total=   0.0s
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:    9.8s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] ............. alpha=1000, score=0.6322619548961357, total=   0.0s
[Parallel(n_jobs=1)]: Done  71 out of  71 | elapsed:   10.0s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] ............. alpha=1000, score=0.6467784518359359, total=   0.0s
[Parallel(n_jobs=1)]: Done  72 out of  72 | elapsed:   10.1s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] ............. alpha=1000, score=0.6481250353717543, total=   0.0s
[Parallel(n_jobs=1)]: Done  73 out of  73 | elapsed:   10.3s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] ............. alpha=1000, score=0.6335037214280215, total=   0.0s
[Parallel(n_jobs=1)]: Done  74 out of  74 | elapsed:   10.4s remaining:
0.0s
[CV] alpha=1000 ......................................................
[CV] .............. alpha=1000, score=0.633677328600126, total=   0.0s
[Parallel(n_jobs=1)]: Done  75 out of  75 | elapsed:   10.5s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.6236867865371746, total=   0.0s
[Parallel(n_jobs=1)]: Done  76 out of  76 | elapsed:   10.7s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.6376768740514698, total=   0.0s
[Parallel(n_jobs=1)]: Done  77 out of  77 | elapsed:   10.8s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............. alpha=10000, score=0.638900711776604, total=   0.0s
[Parallel(n_jobs=1)]: Done  78 out of  78 | elapsed:   11.0s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.6249045669594282, total=   0.0s
[Parallel(n_jobs=1)]: Done  79 out of  79 | elapsed:   11.1s remaining:
0.0s
[CV] alpha=10000 .....................................................
[CV] ............ alpha=10000, score=0.6264462820334563, total=   0.0s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:   11.2s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.6225530561591889, total=   0.0s
[Parallel(n_jobs=1)]: Done  81 out of  81 | elapsed:   11.4s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.6365090833121283, total=   0.0s
```

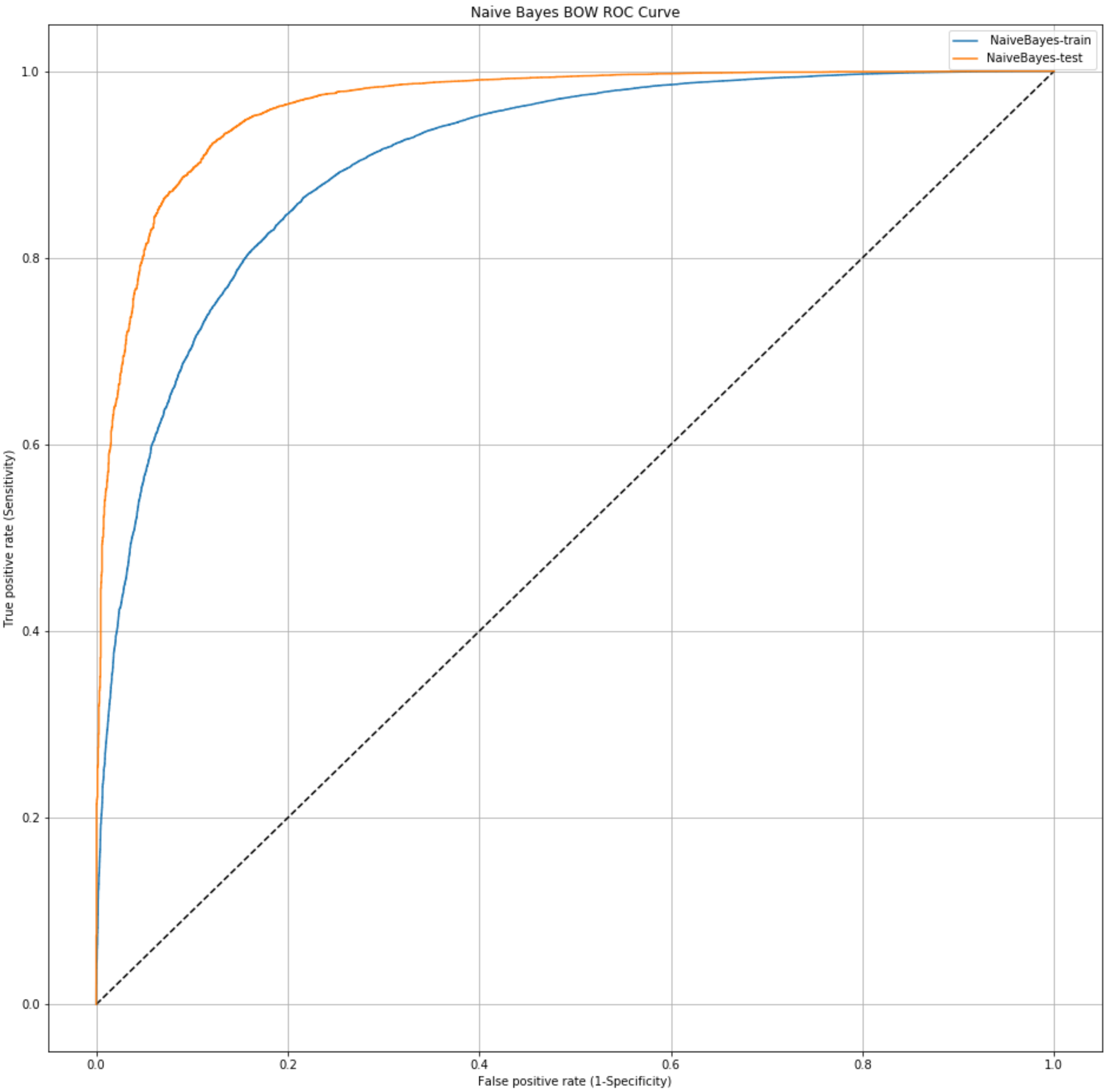
```
[Parallel(n_jobs=1)]: Done  82 out of  82 | elapsed:   11.5s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.6377062124817133, total=   0.0s
[Parallel(n_jobs=1)]: Done  83 out of  83 | elapsed:   11.7s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.6238021299045117, total=   0.0s
[Parallel(n_jobs=1)]: Done  84 out of  84 | elapsed:   11.8s remaining:
0.0s
[CV] alpha=100000 ....................................................
[CV] ........... alpha=100000, score=0.6255166279892576, total=   0.0s
[Parallel(n_jobs=1)]: Done  85 out of  85 | elapsed:   12.0s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.6224394893213256, total=   0.0s
[Parallel(n_jobs=1)]: Done  86 out of  86 | elapsed:   12.1s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.6363900900724561, total=   0.0s
[Parallel(n_jobs=1)]: Done  87 out of  87 | elapsed:   12.2s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] ............ alpha=1000000, score=0.63758735369661, total=   0.0s
[Parallel(n_jobs=1)]: Done  88 out of  88 | elapsed:   12.4s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.6236878623522497, total=   0.0s
[Parallel(n_jobs=1)]: Done  89 out of  89 | elapsed:   12.5s remaining:
0.0s
[CV] alpha=1000000 ...................................................
[CV] .......... alpha=1000000, score=0.6254125897342069, total=   0.0s
[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:   12.7s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.6224254388166413, total=   0.0s
[Parallel(n_jobs=1)]: Done  91 out of  91 | elapsed:   12.8s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.6363785589686115, total=   0.0s
[Parallel(n_jobs=1)]: Done  92 out of  92 | elapsed:   13.0s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.6375730111770866, total=   0.0s
[Parallel(n_jobs=1)]: Done  93 out of  93 | elapsed:   13.1s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.6236785541765881, total=   0.0s
[Parallel(n_jobs=1)]: Done  94 out of  94 | elapsed:   13.3s remaining:
0.0s
[CV] alpha=10000000 ..................................................
[CV] ......... alpha=10000000, score=0.6254037178792795, total=   0.0s
[Parallel(n_jobs=1)]: Done  95 out of  95 | elapsed:   13.5s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.6224239853161566, total=   0.0s
[Parallel(n_jobs=1)]: Done  96 out of  96 | elapsed:   13.7s remaining:
```

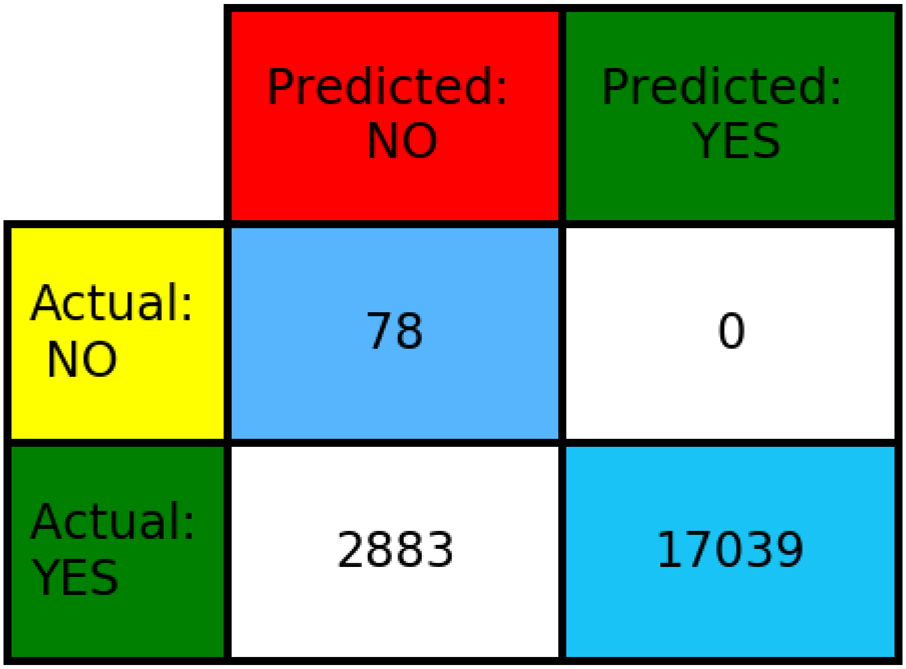
```
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.6363777353183369, total=   0.0s
[Parallel(n_jobs=1)]: Done  97 out of  97 | elapsed:   13.9s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.6375723328146768, total=   0.0s
[Parallel(n_jobs=1)]: Done  98 out of  98 | elapsed:   14.0s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.6236771967343041, total=   0.0s
[Parallel(n_jobs=1)]: Done  99 out of  99 | elapsed:   14.2s remaining:
0.0s
[CV] alpha=100000000 .................................................
[CV] ........ alpha=100000000, score=0.6254026513174851, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.6224238641911164, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.6363774688432481, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.6375723085874478, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.6236771724942634, total=   0.0s
[CV] alpha=1000000000 ................................................
[CV] ....... alpha=1000000000, score=0.6254025301172811, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.6224239610911486, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.6363774930682562, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.6375725266325082, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.6236771967343041, total=   0.0s
[CV] alpha=10000000000 ...............................................
[CV] ...... alpha=10000000000, score=0.6254025058772403, total=   0.0s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:   15.9s finished
0.9412176858633234
{'alpha': 1}
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e),
       fit_params=None, iid='warn', n_jobs=1,
       param_grid={'alpha': [1e-11, 1e-10, 1e-10, 1e-09, 1e-08, 1e-07, 1e-0
6, 1e-05, 0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000, 100000, 1000000, 10
000000, 100000000, 1000000000, 10000000000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=100)
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
```

```
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
```

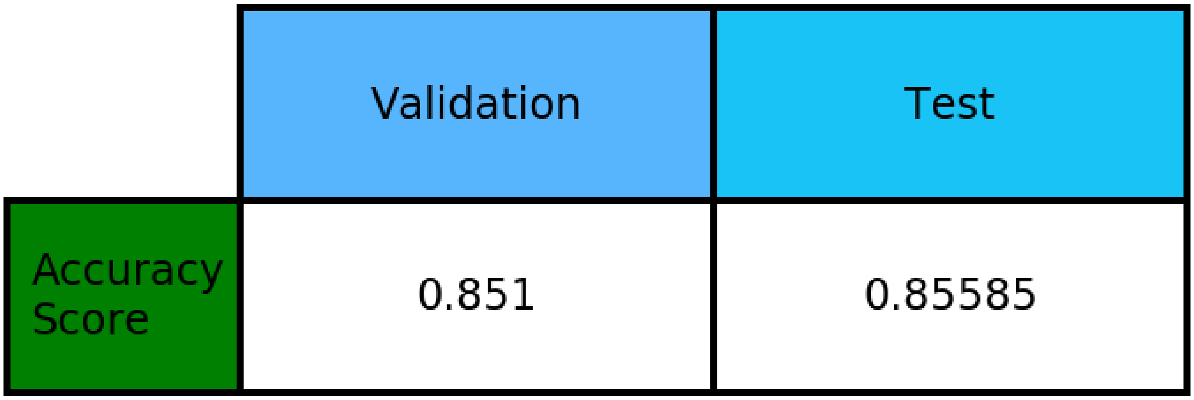
```
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
Function exiting...
[0.9905811497017565, 0.9905811497017565, 0.9905219260665249, 0.990432589507
0133, 0.990295594505154, 0.9900843058363036, 0.9897434214852779, 0.98916968
2462916, 0.9881498441366359, 0.9861671947794589, 0.9817116140799582, 0.9688
370750370262, 0.8357938022724523, 0.6877716758989576, 0.6443691334221521,
0.6336192246041799, 0.6321772935694682, 0.632026502141569, 0.63201178782769
3, 0.6320103901568802]
[0.9997008060000474, 0.9997008060000474, 0.9996999434143802, 0.999698403082
832, 0.9996952299998423, 0.9996894999664825, 0.999675852628964, 0.999642766
3073053, 0.9995496994751536, 0.9991589481679666, 0.9967227597910866, 0.9612
179019550628, 0.7552943043714857, 0.6691463950819801, 0.645046768162603８,
0.6409771505985853, 0.6405307009026097, 0.6404847066025757, 0.6404799623814
068, 0.6404795772985197]
[1e-11, 9.999999999999999e-11, 9.999999999999999e-10, 9.999999999999999e-0
9, 9.999999999999998e-08, 9.999999999999997e-07, 9.999999999999997e-06, 9.9
99999999999998e-05, 0.0009999999999999998, 0.009999999999999998, 0.09999999
999999998, 0.9999999999999998, 9.999999999999998, 99.99999999999999, 999.99
99999999999, 9999.999999999998, 99999.99999999999, 999999.9999999999, 99999
99.999999998, 99999999.99999999]
```


Naive Bayes ROCAUCSCORE plot
NaiveBayes-train
NaiveBayes-val
ROC-AUC-SCORE
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.0
0.2
0.4
0.6
0.8
1.0
1e8
Alpha





	Predicted: NO	Predicted: YES
Actual: NO	78	0
Actual: YES	2883	17039



	Validation	Test
Accuracy Score	0.851	0.85585

[5.2.1] Top 10 important features of positive class from SET 2

In [0]:
```
# Please write all the code with proper documentation
```

In [60]:
```
feature_names = mnbayes_tfidf.tf_idf_vect.get_feature_names()
```

In [61]:
```
top10_posve = np.argsort(mnbayes_tfidf_clf.feature_log_prob_[1])[-10:]

print("%s: %s" % (1,
    " ".join(feature_names[j] for j in top10_posve)))
for k in top10_posve:
    print(" {0} ".format(mnbayes_tfidf_clf.feature_log_prob_[1][k]))
```

```
1: product flavor use like tast good not love great tea
 -6.240321352140638
 -6.217273141048056
 -6.215149461835411
 -6.14808490415411
 -6.1233938511973225
 -6.097917976278151
 -6.043091648591179
 -6.0426606142292085
 -6.026088459211894
 -5.930454976358275
```

[5.2.2] Top 10 important features of negative class from SET 2

In [114]:
```
# Please write all the code with proper documentation
```

In [62]:

```
top10_negve = np.argsort(mnbayes_tfidf_clf.feature_log_prob_[0])[-10:]

print("%s: %s " % (0,
    " ".join(feature_names[j] for j in top10_negve)))
for k in top10_negve:
    print(" {0} ".format(mnbayes_tfidf_clf.feature_log_prob_[0][k]))
```

```
0: tri tea it order one would like product tast not
 -7.188426950174545
 -7.177554369700099
 -7.147491643515288
 -7.131539094498196
 -7.107405544073313
 -6.929602083577947
 -6.751295136319813
 -6.698598341648012
 -6.61501140587509
 -6.201522592294338
```

[6] Conclusions

In [0]:

```
# Please compare all your models using Prettytable library
```

In [63]:

```
from prettytable import from_html_one
L1  =  '<html>'
L2  =  '<head>'

L3  =  '<STYLE TYPE="text/css">'
L4  =  '<!--'
L5  =  'td {font-family: Arial; font-size: 10pt; background-color: #000000;
color: white;}'
L6  =  'THEAD {font-family: Arial; font-size: 14pt; background-color: #0000
00; color: white;}'
L7  =  '--->'
L8  =  '</STYLE>'

L9  =  '</head>'
L10  =  '<body>'

L11  =  '<table border=1 solid> '
L12  =  '<tr>'
L13  =  '<th>Vectorizer </th>'
L15  =  '<th>Hyper Parameter</th>'
L16  =  '<th>AUC</th>'
L17  =  '</tr>'
L18  =  '<tr><td> BOW </td><td> 1 </td><td> 0.88745</td></tr>'
L19  =  '<tr><td> BOW Feat Engg. </td><td> 1 </td><td> 0.88695</td></tr>'
L20  =  '<tr><td> TFIDF </td><td> 1 </td><td> 0.8558</td></tr>'

L22  =  '</table>'

L23  =  '</body>'
L24  =  '</html>'

html_string = L1+L2+L3+L4+L5+L6+L7+L8+L9+L10+L11+L12+L13+L15+L16+L17+L18+L1
9+L20+L22+L23+L24
#html_string = L1+L2+L3+L4+L5+L6+L7+L8+L9+L10+L11+L12+L13+L14+L15+L16+L17+L
18+L22+L23+L24
tbl = from_html_one(html_string)

print(tbl)
```

```
+----------------+-----------------+---------+
|   Vectorizer   | Hyper Parameter |   AUC   |
+----------------+-----------------+---------+
|      BOW       |        1        | 0.88745 |
| BOW Feat Engg. |        1        | 0.88695 |
|     TFIDF      |        1        |  0.8558 |
+----------------+-----------------+---------+
```