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 [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
from bs4 import BeautifulSoup
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

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
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 [62]:
```
# 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 10000""", 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 (10000, 10)
```

Out[62]:

	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 [63]:

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

In [64]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[64]:

	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 [65]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[65]:

	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 [66]:
```
display['COUNT(*)'].sum()
```

Out[66]: 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 [67]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[67]:

	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 [68]:
```
#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 [69]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first', inplace=False)
final.shape
```

Out[69]: (9564, 10)

In [70]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[70]: 95.64

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 [71]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[71]:

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

In [72]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [73]:
```
#Before starting the next phase of preprocessing lets see the number of ent
ries left
print(final.shape)

#How many positive and negative reviews are present in our dataset?
final['Score'].value_counts()
```

```
(9564, 10)
```

Out[73]:
```
1    7976
0    1588
Name: Score, dtype: int64
```

[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 [0]:
```
# 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 genocide. 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 the bsg (my favorite), so I bought some more through amazon and shared with family 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.  They 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 were ordering; the other wants crispy cookies.  Hey, I'm sorry; but these reviews do nobody any good beyond reminding us to look  before ordering.<br /><br />These are chocolate-oatmeal cookies.  If you don't like that combination, 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 differ; so, I've 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 don't see where these taste like raw cookie dough.  Both are soft, however, so is this the confusion?  And, yes, they stick together.  Soft cookies 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 Snaps.  If you want a cookie that's soft, chewy and tastes like a combination 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 [0]:
```
# 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 [0]:
```
# 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 [28]:

```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 [0]:
```
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 [0]:
```
#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 [0]:
```
#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 [74]:

```
# 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 [75]:

```
# 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%|██████████| 9564/9564 [00:04<00:00, 2307.78it/s]
```

In [30]:
```
preprocessed_reviews[1500]
```

Out[30]:
```
'wow far two two star reviews one obviously no idea ordering wants crispy c
ookies hey sorry reviews nobody good beyond reminding us look ordering choc
olate oatmeal cookies not like combination not order type cookie find combo
quite nice really oatmeal sort calms rich chocolate flavor gives cookie sor
t coconut type consistency let also remember tastes differ given opinion so
ft chewy cookies advertised not crispy cookies blurb would say crispy rathe
r chewy happen like raw cookie dough however not see taste like raw cookie 
dough soft however confusion yes stick together soft cookies tend not indiv
idually wrapped would add cost oh yeah chocolate chip cookies tend somewhat
sweet want something hard crisp suggest nabiso ginger snaps want cookie sof
t chewy tastes like combination chocolate oatmeal give try place second ord
er'
```

[3.2] Preprocessing Review Summary

In [0]:
```
## Similartly you can do preprocessing for review summary also.
```

[4] Featurization

[4.1] BAG OF WORDS

In [0]:
```
#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 [0]:
```
#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.ht
ml

# 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=500
0)
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 ", fi
nal_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 [0]:
```
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_fe
ature_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 ", fi
nal_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 [0]:
```
# 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 [0]:
```
# 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 ")
```

```
[('snack', 0.9951335191726685), ('calorie', 0.9946465492248535), ('wonderfu
l', 0.9946032166481018), ('excellent', 0.9944332838058472), ('especially',
0.9941144585609436), ('baked', 0.9940600395202637), ('salted', 0.9940472245
21637), ('alternative', 0.9937226176261902), ('tasty', 0.9936816692352295),
('healthy', 0.9936649799346924)]
==================================================
[('varieties', 0.9994194507598877), ('become', 0.9992934465408325), ('popco
rn', 0.9992750883102417), ('de', 0.9992610216140747), ('miss', 0.9992451071
739197), ('melitta', 0.999218761920929), ('choice', 0.9992102384567261),
('american', 0.9991837739944458), ('beef', 0.9991780519485474), ('finish',
0.9991567134857178)]
```

In [0]:
```
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 [0]:
```
# 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 [00:03<00:00, 1330.47it/s]

4986
50
```

[4.4.1.2] TFIDF weighted W2v

In [0]:
```
# 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 [0]:
```
# 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 [00:20<00:00, 245.63it/s]
```

[5] Assignment 5: Apply Logistic Regression

1. **Apply Logistic Regression 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)
   - SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
   - SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)

2. **Hyper paramter tuning (find best hyper parameters corresponding the algorithm that you choose)**

   - 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
   - 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. **Pertubation Test**

   - Get the weights W after fit your model with the data X i.e Train data.
   - Add a noise to the X (X' = X + e) and get the new data set X' (if X is a sparse matrix, X.data+=e)
   - Fit the model again on data X' and get the weights W'
   - Add a small eps value(to eliminate the divisible by zero error) to W and W' i.e W=W+10^-6 and W' = W'+10^-6
   - Now find the % change between W and W' (| (W-W') / (W) |)*100)
   - Calculate the 0th, 10th, 20th, 30th, ...100th percentiles, and observe any sudden rise in the values of percentage_change_vector
   - Ex: consider your 99th percentile is 1.3 and your 100th percentiles are 34.6, there is sudden rise from 1.3 to 34.6, now calculate the 99.1, 99.2, 99.3,..., 100th percentile values and get the proper value after which there is sudden rise the values, assume it is 2.5
   - Print the feature names whose % change is more than a threshold x(in our example it's 2.5)

4. **Sparsity**

   - Calculate sparsity on weight vector obtained after using L1 regularization

   NOTE: Do sparsity and multicollinearity for any one of the vectorizers. Bow or tf-idf is recommended.

5. **Feature importance**

   - Get top 10 important features for both positive and negative classes separately.

6. **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.

7. **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.
     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)
8. **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 Logistic Regression

[5.0] Logistic Regression - classes

In [2]:
```
# the required imports
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import classification_report,accuracy_score,confusion_
matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score

from sklearn.preprocessing import StandardScaler

from nltk.stem.porter import PorterStemmer
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

from tqdm import tqdm
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
import re
import string
import pickle

class wordvect(enum.Enum):
    BOW = 1
    TFIDF = 2
    W2VAVG = 3
    TFIDFAVG = 4

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

class LogisticRegrsn:
    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.log_regr = None
        self.logrgr_lambda = []
        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
        self.sentlist= []

    def logRegrsn(self):
        self.log_regr = LogisticRegression(max_iter=200,random_state=42)
        return self.log_regr

    def getlogRegresion(self):
        return self.log_regr

    @property
    def log_regr(self):
        return self._log_regr

    @log_regr.setter
    def log_regr(self,new_mnbclf):
        self._log_regr = 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 logrgr_lambda (self):
        return self._logrgr_lambda

    @logrgr_lambda.setter
    def logrgr_lambda (self,new_logrgr_lambda):
        self._logrgr_lambda = new_logrgr_lambda


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

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

```
    def set_lambdaparm(self,prmval):
        print(prmval)
        params = {'C':prmval}
        (self.log_regr).set_params(**params)
        return self.log_regr

    def set_penaltyparm(self,prmval):
        params = {'penalty':prmval}
        (self.log_regr).set_params(**params)
        return self.log_regr

    def logRegr_fitdata(self,x_data,y_data):
        self.log_regr.fit(x_data,y_data)
        return self.log_regr

    def logRegr_predict(self,x_data):
        self.predicted = self.log_regr.predict(x_data)
        return [self.predicted,self.log_regr]

    def hyperparamtuning(self,typevect,hyperparam,measure,cvfold=5,vbose=0,
njob=1):

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

        #initialize the classifier
        grdsch_clf = self.getlogRegresion()
        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 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 listsent(self,xdata):
        self.sentlist = []
        for sentance in xdata :
            self.sentlist.append(sentance.split())
        return self.sentlist

    # average Word2Vec
    # compute average word2vec for each review.
    def w2vec_crea(self,list_of_sentance,w2v_model,w2v_words):
        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 5
0, you might need to change this to 300 if you use google's w2v
            cnt_words =0; # num of words with a valid vector in the sentenc
e/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)
        return sent_vectors
        #print(sent_vectors[0])
        #print(len(sent_vectors[0]))
        return sent_vectors

    def tfidfwtw2v_crea(self,tfidf_feat, list_of_sentance, w2v_model,w2v_wo
rds,diction):
        tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review i
s 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 senten
ce/review
            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
                    print(diction[word],sent.count(word),len(sent))
                    denom = sent.count(word)/len(sent)
                    tf_idf = diction[word]*(denom)
                    #tf_idf = 1
                    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
        return tfidf_sent_vectors

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

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

            # fit the x-train model
            (self.log_regr).fit(self.xtrain,self.ytrain)
            self.yprdprobatrn =  (self.log_regr).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.log_regr).fit(self.xval,self.yval)
            self.yprdprobaval =  (self.log_regr).predict_proba(self.xval)
[:,1]
            (self.rocaucscoreval).append(roc_auc_score(self.yval,self.yprdp
robaval))
            print('Fitting probability generation and roc auc score generat
ion for validation data complete...')

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

            # calculate accuracy_score
            self.accuracy_score_val = accuracy_score(self.yval, self.predic
ted)

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

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

        print('Function exiting...')

    def actualClasifier_logregrsn(self,parm_lambda):
        self.set_lambdaparm(parm_lambda)
        (self.log_regr).fit(self.xtest,self.ytest)
```

```
        # predict xtest labels
        self.test_predict = (self.log_regr).predict(self.xtest)

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

        # calculate accuracy_score
        self.accuracy_score_test = accuracy_score(self.ytest, self.test_pre
dict)

        # 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 ).r
avel()
        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.log_regr).predict_proba(self.xtra
in)[:,1]
        fpr_trn, tpr_trn, thrshld_trn = roc_curve(self.ytrain, self.ytrn_pr
edprob_actclf)

        # predict probabilites  from xtest for roc_curve
        self.ytst_predprob_actclf = (self.log_regr).predict_proba(self.xtes
t)[:,1]
        fpr, tpr, thrshld_test = roc_curve(self.ytest,self.ytst_predprob_ac
tclf)

        # 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 load_data(self):

        with open ('E:/appliedaiacourse/assignments/dblite/preproc_xtrain',
'rb') 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') a
```

```
s 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
]

    def alt_load_data(self):
        with open ('E:/appliedaiacourse/assignments/dblite/alt_preproc/ppro
c_xtrain', 'rb') as fp:
            xtrain_preproc = pickle.load(fp)

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

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

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

        with open ('E:/appliedaiacourse/assignments/dblite/alt_preproc/ytes
t', 'rb') as fp:
            ytest = pickle.load(fp)

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

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

    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['logrgr_lambda'] = self.logrgr_lambda
        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_p
arameters['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()

    def draw_sparsity(self,data):
        #colors = [["#56b5fd","w"]]
        table = plt.table(cellText=data,colLabels=['Lambda','Non-Zero \n Co
lumns'], rowLabels=['1','2','3','4'], loc='center',
                          cellLoc='center',colColours=["#56b5fd","#1ac3f5"
])

        table.set_fontsize(20)
        for i in range(0,5):
            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(0.4)
                table.get_celld()[(i,j)].set_linewidth(4)
        plt.axis('off')
        plt.show()
```

[5.1] Logistic Regression on BOW, SET 1

In [77]:
```
logregr = LogisticRegrsn()
log_regr = logregr.logRegrsn()

logregr.xtrain,logregr.xtest,logregr.xval, logregr.ytrain,logregr.ytest,log
regr.yval = logregr.load_data()

# vectorise the complete corpus
logregr.BOWVectorizer()

# hyperparameter tuning for lambda
print(logregr.getlogRegresion())
return_63 = logregr.hyperparamtuning(wordvect.BOW,[0.00000000001,0.00000000
01,0.0000000001,0.000000001,0.00000001,0.0000001,0.000001,0.00001,0.0001,0.
001,0.01,1,10,100,1000,10000,100000,1000000,10000000,100000000,1000000000,1
0000000000],'roc_auc',5,100,1)
#output parameter 0.9211633325857863 {'C': 1}

print(return_63[0])
print(return_63[1])
print(return_63[2])

logregr.calcrocaucscore_logregrsn(10000000000)
print(logregr.rocaucscoretrn)
print(logregr.rocaucscoreval)
print( logregr.logrgr_lambda)

# testing code for displayig graphs
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr ROCAUCSCORE plot'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-val'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='C'
displaygraph.label_y='ROC-AUC-SCORE'
displaygraph.Xdata = logregr.logrgr_lambda
displaygraph.ydatatrn = logregr.rocaucscoretrn
displaygraph.ydataval = logregr.rocaucscoreval
displaygraph.rocacuscoregraph()
```

```
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
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
Fitting 5 folds for each of 22 candidates, totalling 110 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[CV] C=1e-11 ..........................................................
[CV] ................ C=1e-11, score=0.4908855345112372, total=   0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:
0.0s
[CV] C=1e-11 ..........................................................
[CV] ................ C=1e-11, score=0.5015941024314737, total=   0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.3s remaining:
0.0s
[CV] C=1e-11 ..........................................................
[CV] ................ C=1e-11, score=0.5034982664933161, total=   0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.4s remaining:
0.0s
[CV] C=1e-11 ..........................................................
[CV] ............... C=1e-11, score=0.48785690308851093, total=   0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.6s remaining:
0.0s
[CV] C=1e-11 ..........................................................
[CV] ................ C=1e-11, score=0.4872406485316256, total=   0.1s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.8s remaining:
0.0s
[CV] C=1e-10 ..........................................................
[CV] ................. C=1e-10, score=0.490886091686423, total=   0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    1.0s remaining:
0.0s
[CV] C=1e-10 ..........................................................
[CV] ................ C=1e-10, score=0.5015946838316676, total=   0.1s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    1.2s remaining:
0.0s
[CV] C=1e-10 ..........................................................
[CV] .................. C=1e-10, score=0.50349795153934, total=   0.1s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    1.4s remaining:
0.0s
[CV] C=1e-10 ..........................................................
[CV] ............... C=1e-10, score=0.48785685460842926, total=   0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.6s remaining:
0.0s
[CV] C=1e-10 ..........................................................
[CV] ................ C=1e-10, score=0.4872413272527676, total=   0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    1.8s remaining:
0.0s
[CV] C=1e-10 ..........................................................
[CV] ................. C=1e-10, score=0.490886091686423, total=   0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    1.9s remaining:
```

```
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5015946838316676, total=   0.1s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    2.1s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] .................. C=1e-10, score=0.50349795153934, total=   0.1s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    2.4s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ............... C=1e-10, score=0.48785685460842926, total=   0.0s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    2.6s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.4872413272527676, total=   0.0s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:    2.8s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................ C=1e-09, score=0.4908858736613503, total=   0.1s
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed:    3.0s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................ C=1e-09, score=0.5015991896831699, total=   0.1s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed:    3.2s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................ C=1e-09, score=0.5035021913044018, total=   0.1s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    3.4s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ............... C=1e-09, score=0.48786126629585225, total=   0.1s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed:    3.6s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................. C=1e-09, score=0.487244890538763, total=   0.1s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    3.8s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ............... C=1e-08, score=0.49092230807349746, total=   0.1s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    4.1s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ................ C=1e-08, score=0.5016355756453009, total=   0.1s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed:    4.3s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ................. C=1e-08, score=0.503536739332848, total=   0.1s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:    4.5s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ............... C=1e-08, score=0.48790135932331113, total=   0.1s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:    4.7s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ............... C=1e-08, score=0.48727562691047877, total=   0.1s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    4.9s remaining:
0.0s
```

```
[CV] C=1e-07 ..........................................................
[CV] ............... C=1e-07, score=0.49124866738230577, total=   0.1s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    5.2s remaining:
0.0s
[CV] C=1e-07 ..........................................................
[CV] ................ C=1e-07, score=0.501990375113591, total=   0.1s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    5.4s remaining:
0.0s
[CV] C=1e-07 ..........................................................
[CV] ................ C=1e-07, score=0.503882413435142, total=   0.1s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    5.6s remaining:
0.0s
[CV] C=1e-07 ..........................................................
[CV] ............... C=1e-07, score=0.4882597240862813, total=   0.1s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    5.8s remaining:
0.0s
[CV] C=1e-07 ..........................................................
[CV] ............... C=1e-07, score=0.4876296284661077, total=   0.1s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    6.0s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ............... C=1e-06, score=0.4945631814373647, total=   0.1s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    6.3s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ............... C=1e-06, score=0.5055196196402412, total=   0.1s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    6.5s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ............... C=1e-06, score=0.5073945863578606, total=   0.1s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    6.7s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] .............. C=1e-06, score=0.49184104467206813, total=   0.2s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    7.1s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ............... C=1e-06, score=0.4910625515222067, total=   0.1s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    7.3s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ................ C=1e-05, score=0.524330096161654, total=   0.2s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    7.6s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ............... C=1e-05, score=0.5371771508948621, total=   0.1s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:    7.9s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ............... C=1e-05, score=0.5384871820515384, total=   0.1s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:    8.2s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ............... C=1e-05, score=0.5240368122833788, total=   0.1s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:    8.4s remaining:
0.0s
[CV] C=1e-05 ..........................................................
```

```
[CV] ................ C=1e-05, score=0.5217300330420843, total=   0.1s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    8.7s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ............... C=0.0001, score=0.7153462228464307, total=   0.2s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:    9.0s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ............... C=0.0001, score=0.7325623062362728, total=   0.2s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    9.3s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ................. C=0.0001, score=0.72913483208786, total=   0.2s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:    9.7s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ............... C=0.0001, score=0.7280428862558386, total=   0.2s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:   10.0s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ............... C=0.0001, score=0.7100247791392924, total=   0.2s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:   10.3s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.8875047795940936, total=   0.4s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:   10.8s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................. C=0.001, score=0.886567804731706, total=   0.4s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:   11.3s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.8947453369367944, total=   0.4s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:   11.8s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.8908863167296811, total=   0.3s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:   12.3s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.8827328881310482, total=   0.4s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:   12.8s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] .................. C=0.01, score=0.916464856199383, total=   0.6s
[Parallel(n_jobs=1)]: Done  51 out of  51 | elapsed:   13.5s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] ................. C=0.01, score=0.9163852043728271, total=   0.6s
[Parallel(n_jobs=1)]: Done  52 out of  52 | elapsed:   14.2s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] ................. C=0.01, score=0.9260396969929552, total=   0.6s
[Parallel(n_jobs=1)]: Done  53 out of  53 | elapsed:   14.9s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] .................. C=0.01, score=0.918223701446471, total=   0.6s
```

```
[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:   15.7s remaining:
0.0s
[CV] C=0.01 ..........................................................
[CV] ................. C=0.01, score=0.9146042755165698, total=   0.6s
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:   16.4s remaining:
0.0s
[CV] C=1 .............................................................
[CV] ................... C=1, score=0.9187237897525113, total=   1.3s
[Parallel(n_jobs=1)]: Done  56 out of  56 | elapsed:   17.8s remaining:
0.0s
[CV] C=1 .............................................................
[CV] ................... C=1, score=0.9192586779308433, total=   1.9s
[Parallel(n_jobs=1)]: Done  57 out of  57 | elapsed:   19.9s remaining:
0.0s
[CV] C=1 .............................................................
[CV] ................... C=1, score=0.9290699903323665, total=   1.3s
[Parallel(n_jobs=1)]: Done  58 out of  58 | elapsed:   21.4s remaining:
0.0s
[CV] C=1 .............................................................
[CV] ................... C=1, score=0.9216994808946786, total=   1.4s
[Parallel(n_jobs=1)]: Done  59 out of  59 | elapsed:   22.9s remaining:
0.0s
[CV] C=1 .............................................................
[CV] ................... C=1, score=0.9170647850965248, total=   1.4s
[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:   24.4s remaining:
0.0s
[CV] C=10 ............................................................
[CV] .................. C=10, score=0.9182671968002835, total=   1.4s
[Parallel(n_jobs=1)]: Done  61 out of  61 | elapsed:   25.9s remaining:
0.0s
[CV] C=10 ............................................................
[CV] ................... C=10, score=0.918811484281749, total=   1.7s
[Parallel(n_jobs=1)]: Done  62 out of  62 | elapsed:   27.8s remaining:
0.0s
[CV] C=10 ............................................................
[CV] .................. C=10, score=0.9285767724059227, total=   1.9s
[Parallel(n_jobs=1)]: Done  63 out of  63 | elapsed:   29.9s remaining:
0.0s
[CV] C=10 ............................................................
[CV] .................. C=10, score=0.9214264410752728, total=   1.3s
[Parallel(n_jobs=1)]: Done  64 out of  64 | elapsed:   31.3s remaining:
0.0s
[CV] C=10 ............................................................
[CV] .................. C=10, score=0.9165679612205889, total=   1.9s
[Parallel(n_jobs=1)]: Done  65 out of  65 | elapsed:   33.4s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] ................. C=100, score=0.9182122060319494, total=   1.4s
[Parallel(n_jobs=1)]: Done  66 out of  66 | elapsed:   34.9s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] ................. C=100, score=0.9187607086648203, total=   1.8s
[Parallel(n_jobs=1)]: Done  67 out of  67 | elapsed:   36.8s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] ................. C=100, score=0.9285105836165012, total=   2.1s
[Parallel(n_jobs=1)]: Done  68 out of  68 | elapsed:   39.0s remaining:
```

```
0.0s
[CV] C=100 ...........................................................
[CV] .................. C=100, score=0.9213930382990705, total=   1.3s
[Parallel(n_jobs=1)]: Done  69 out of  69 | elapsed:   40.5s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] .................. C=100, score=0.9164940775762752, total=   2.0s
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:   42.6s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9182058106298173, total=   1.4s
[Parallel(n_jobs=1)]: Done  71 out of  71 | elapsed:   44.2s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] .................. C=1000, score=0.918755524513092, total=   1.7s
[Parallel(n_jobs=1)]: Done  72 out of  72 | elapsed:   46.0s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9285036546290288, total=   2.0s
[Parallel(n_jobs=1)]: Done  73 out of  73 | elapsed:   48.2s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9213903234145026, total=   1.3s
[Parallel(n_jobs=1)]: Done  74 out of  74 | elapsed:   49.7s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9164857390022451, total=   1.9s
[Parallel(n_jobs=1)]: Done  75 out of  75 | elapsed:   51.7s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9182052292296233, total=   1.4s
[Parallel(n_jobs=1)]: Done  76 out of  76 | elapsed:   53.3s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9187549915629143, total=   1.8s
[Parallel(n_jobs=1)]: Done  77 out of  77 | elapsed:   55.2s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9285027824487875, total=   2.1s
[Parallel(n_jobs=1)]: Done  78 out of  78 | elapsed:   57.4s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9213887235718108, total=   1.4s
[Parallel(n_jobs=1)]: Done  79 out of  79 | elapsed:   58.9s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9164848663607768, total=   2.0s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:  1.0min remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.9182052776796396, total=   1.4s
[Parallel(n_jobs=1)]: Done  81 out of  81 | elapsed:  1.0min remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.9187547493128335, total=   1.8s
[Parallel(n_jobs=1)]: Done  82 out of  82 | elapsed:  1.1min remaining:
0.0s
```

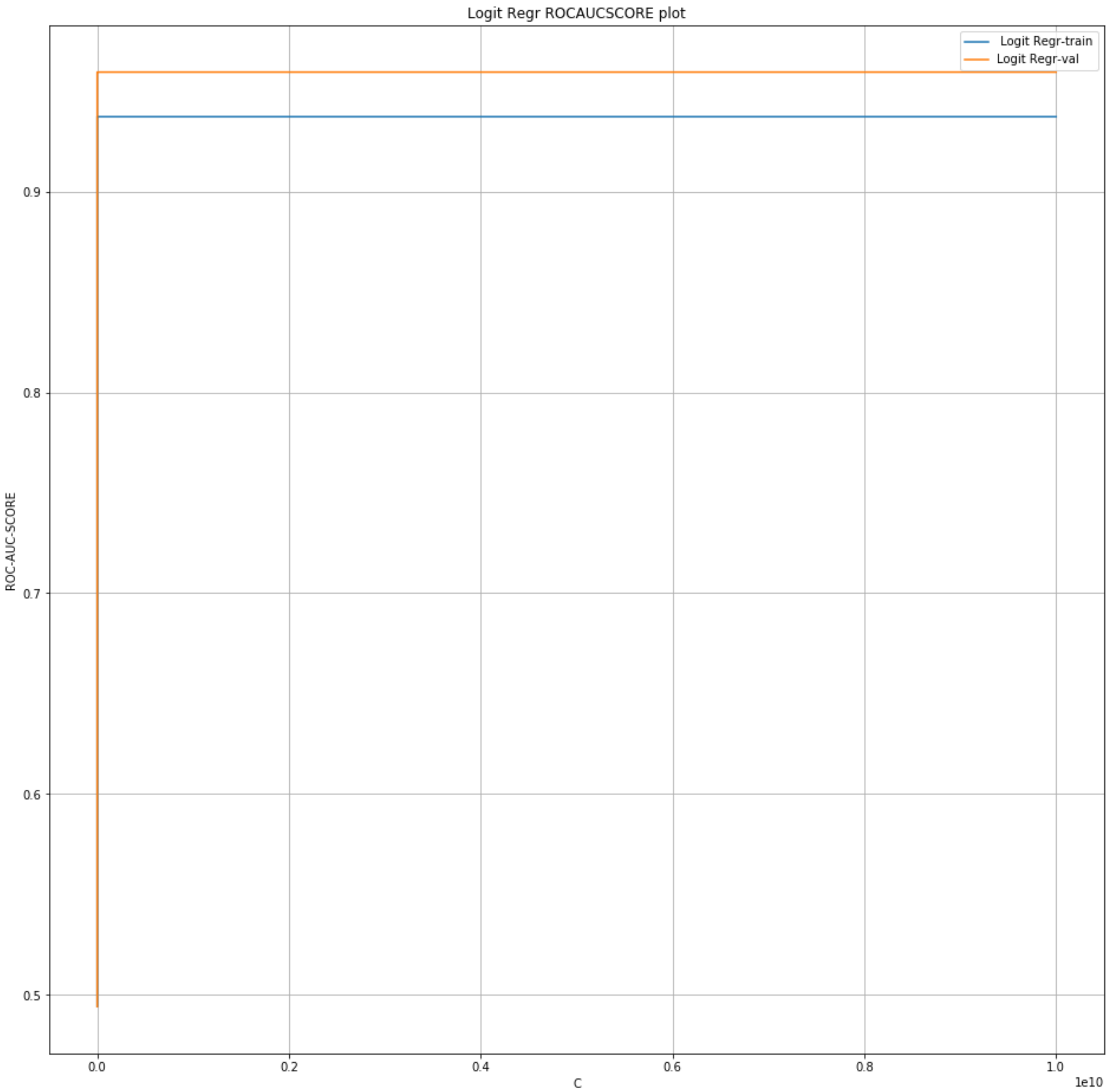
```
[CV] C=100000 .........................................................
[CV] ............... C=100000, score=0.9285033154478239, total=   2.1s
[Parallel(n_jobs=1)]: Done  83 out of  83 | elapsed:  1.1min remaining:
0.0s
[CV] C=100000 .........................................................
[CV] ............... C=100000, score=0.9213894507730342, total=   1.4s
[Parallel(n_jobs=1)]: Done  84 out of  84 | elapsed:  1.1min remaining:
0.0s
[CV] C=100000 .........................................................
[CV] ............... C=100000, score=0.9164845270002058, total=   2.0s
[Parallel(n_jobs=1)]: Done  85 out of  85 | elapsed:  1.2min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9182052292296234, total=   1.5s
[Parallel(n_jobs=1)]: Done  86 out of  86 | elapsed:  1.2min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9187556698631403, total=   1.8s
[Parallel(n_jobs=1)]: Done  87 out of  87 | elapsed:  1.2min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9285032669933659, total=   1.9s
[Parallel(n_jobs=1)]: Done  88 out of  88 | elapsed:  1.3min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9213887720518923, total=   1.3s
[Parallel(n_jobs=1)]: Done  89 out of  89 | elapsed:  1.3min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9164842361197164, total=   2.0s
[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:  1.3min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9182051807796072, total=   1.4s
[Parallel(n_jobs=1)]: Done  91 out of  91 | elapsed:  1.4min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9187556698631404, total=   1.8s
[Parallel(n_jobs=1)]: Done  92 out of  92 | elapsed:  1.4min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9285030731755346, total=   2.0s
[Parallel(n_jobs=1)]: Done  93 out of  93 | elapsed:  1.4min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9213888205319738, total=   1.4s
[Parallel(n_jobs=1)]: Done  94 out of  94 | elapsed:  1.4min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9164845270002058, total=   2.0s
[Parallel(n_jobs=1)]: Done  95 out of  95 | elapsed:  1.5min remaining:
0.0s
[CV] C=100000000 ......................................................
[CV] ............ C=100000000, score=0.9182052776796396, total=   1.4s
[Parallel(n_jobs=1)]: Done  96 out of  96 | elapsed:  1.5min remaining:
0.0s
[CV] C=100000000 ......................................................
```

```
[CV] ........... C=100000000, score=0.9187557183131564, total=   1.8s
[Parallel(n_jobs=1)]: Done  97 out of  97 | elapsed:  1.5min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ........... C=100000000, score=0.9285028793577031, total=   2.1s
[Parallel(n_jobs=1)]: Done  98 out of  98 | elapsed:  1.6min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ........... C=100000000, score=0.9213894022929526, total=   1.4s
[Parallel(n_jobs=1)]: Done  99 out of  99 | elapsed:  1.6min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ........... C=100000000, score=0.9164843330798795, total=   1.9s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9182053261296557, total=   1.4s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9187547977628496, total=   1.7s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9285026855398717, total=   2.1s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9213881418108318, total=   1.4s
[CV] C=1000000000 ....................................................
[CV] ........... C=1000000000, score=0.916484720920532, total=   1.9s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9182053261296557, total=   1.4s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9187547493128335, total=   1.7s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9285031700844504, total=   1.9s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9213881418108318, total=   1.4s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9164846239603689, total=   2.0s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:  1.9min finished
0.9211633325857863
{'C': 1}
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=1,
       param_grid={'C': [1e-11, 1e-10, 1e-10, 1e-09, 1e-08, 1e-07, 1e-06, 1
e-05, 0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000, 100000, 1000000, 100000
00, 100000000, 1000000000, 10000000000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=100)
1e-11
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...
9.999999999999999e-11
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...
9.999999999999999e-10
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...
9.999999999999999e-09
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...
9.999999999999998e-08
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...
9.999999999999997e-07
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...
9.999999999999997e-06
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...
9.999999999999998e-05
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...
0.0009999999999999998
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...
0.009999999999999998
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...
0.09999999999999998
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...
```

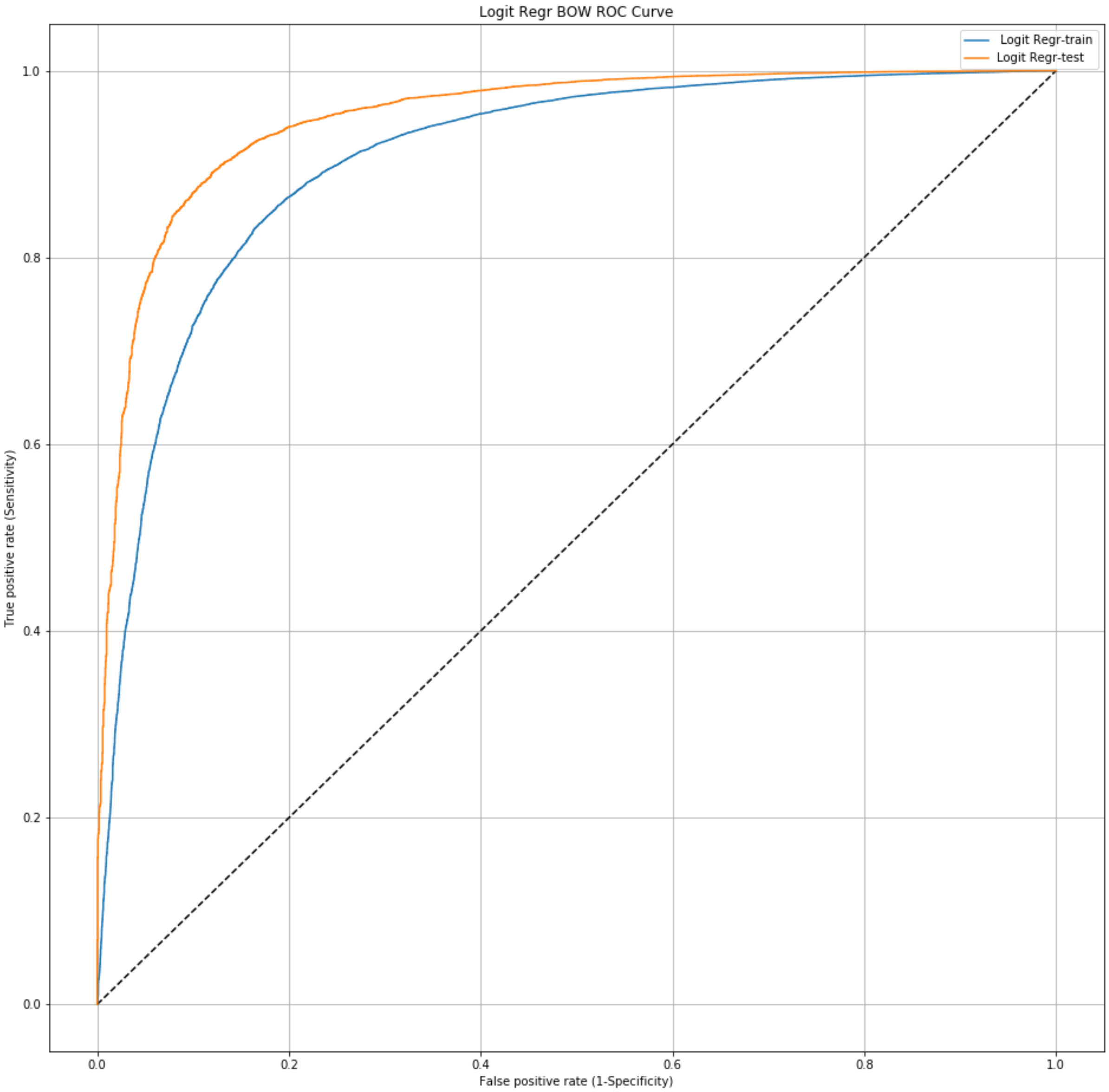
```
0.9999999999999998
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
9.999999999999998
Fitting probability generation 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...
99.99999999999999
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...
999.9999999999999
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...
9999.999999999998
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...
99999.99999999999
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...
999999.9999999999
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...
9999999.999999998
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...
99999999.99999999
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...
999999999.9999999
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...
9999999999.999998
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...
[0.49441245811572154, 0.49441245811572154, 0.4944173131317419, 0.4944607940
6978777, 0.4948995754717388, 0.4992622527403052, 0.5372666222198301, 0.7554
104922733074, 0.8981360064977231, 0.928681348028337, 0.9365707965550443, 0.
9372419693227502, 0.9372380283175319, 0.9372365996304457, 0.937235473352023
8, 0.937235824223615, 0.9372350895256425, 0.9372350662634374, 0.93723620804
99961, 0.9372358377932344, 0.9372358203465807, 0.9372358145310294]
[0.494159925354301, 0.494159925354301, 0.494159925354301, 0.494173233818878
73, 0.4942875572263976, 0.4954205326934755, 0.5064495838393835, 0.591727051
770913, 0.8523730840740069, 0.9275395076558174, 0.9515744714567939, 0.95856
89321780929, 0.959413958065521, 0.9594502174701695, 0.9594528668404325, 0.9
594530208735874, 0.9594520658680273, 0.9594528052271705, 0.959451850221610
6, 0.9594521274812894, 0.9594532057133731, 0.9594531132934803]
[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, 999999999.9999999, 9999999999.999998]
```


Logit Regr ROCAUCSCORE plot
Logit Regr-train
Logit Regr-val
ROC-AUC-SCORE
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0
1e10
C

In [78]:
```
#process test data using logistic regression
logregr.actualClasifier_logregrsn(1)

# display ROC AUC graph for test data
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr BOW ROC Curve'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-test'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='False positive rate (1-Specificity)'
displaygraph.label_y='True positive rate (Sensitivity)'
displaygraph.constructgraph(logregr.roc_curve_test['fpr_trn'],logregr.roc_c
urve_test['tpr_trn'],\
                            logregr.roc_curve_test['fpr'],logregr.roc_curve
_test['tpr'])
data = [[logregr.confsnmtxytstpred['tn'] ,logregr.confsnmtxytstpred['fn']],
[logregr.confsnmtxytstpred['fp'],logregr.confsnmtxytstpred['tp']]]
displaygraph.draw_table(data)

data1= [[logregr.accuracy_score_val,logregr.accuracy_score_test]]
displaygraph.draw_accscore(data1)
```

1


Logit Regr BOW ROC Curve
Logit Regr-train
Logit Regr-test
True positive rate (Sensitivity)
False positive rate (1-Specificity)
0.0
0.2
0.4
0.6
0.8
1.0

	Predicted: NO	Predicted: YES
Actual: NO	1828	402
Actual: YES	1133	16637

	Validation	Test
Accuracy Score	0.93175	0.92325

[5.1.1] Applying Logistic Regression with L1 regularization on BOW, SET 1

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

In [4]:
```
#instantiate logistic regression object
lgrgr_l1 = LogisticRegrsn()
#instantiate logistic regression classifier
lgrgrl1_clf = lgrgr_l1.logRegrsn()

#load the data
lgrgr_l1.xtrain,lgrgr_l1.xtest,lgrgr_l1.xval, lgrgr_l1.ytrain,lgrgr_l1.ytes
t,lgrgr_l1.yval = lgrgr_l1.load_data()

# vectorise the training corpus
lgrgr_l1.BOWVectorizer()

# print the shapes of the data vetors
print((lgrgr_l1.xtrain).shape)
print((lgrgr_l1.xtest).shape)
print((lgrgr_l1.xval).shape)
print((lgrgr_l1.ytrain).shape)
print((lgrgr_l1.ytest).shape)
print((lgrgr_l1.yval).shape)
```

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

In [84]:
```
#set lambda paramater
lgrgr_l1.set_lambdaparm(10)
#set penalty
lgrgr_l1.set_penaltyparm('l1')
print(lgrgr_l1.getlogRegresion())
#fit test data
lgrgr_l1.logRegr_fitdata(lgrgr_l1.xtest,lgrgr_l1.ytest)
lone_tst_1= lgrgr_l1.getlogRegresion()
#get coefficients
w_1 = lone_tst_1.coef_
print(np.count_nonzero(w_1))

#set lambda paramater
lgrgr_l1.set_lambdaparm(1)
#set penalty
lgrgr_l1.set_penaltyparm('l1')
print(lgrgr_l1.getlogRegresion())
#fit test data
lgrgr_l1.logRegr_fitdata(lgrgr_l1.xtest,lgrgr_l1.ytest)
lone_tst_2 = lgrgr_l1.getlogRegresion()
#get coefficients
w_2 = lone_tst_2.coef_
print(np.count_nonzero(w_2))

#set lambda paramater
lgrgr_l1.set_lambdaparm(0.1)
#set penalty
lgrgr_l1.set_penaltyparm('l1')
print(lgrgr_l1.getlogRegresion())
#fit test data
lgrgr_l1.logRegr_fitdata(lgrgr_l1.xtest,lgrgr_l1.ytest)
lone_tst_3 = lgrgr_l1.getlogRegresion()
#get coefficients
w_3 = lone_tst_3.coef_
print(np.count_nonzero(w_3))

#set lambda paramater
lgrgr_l1.set_lambdaparm(0.001)
#set penalty
lgrgr_l1.set_penaltyparm('l1')
print(lgrgr_l1.getlogRegresion())
#fit test data
lgrgr_l1.logRegr_fitdata(lgrgr_l1.xtest,lgrgr_l1.ytest)
lone_tst_4 = lgrgr_l1.getlogRegresion()
#get coefficients
w_4 = lone_tst_4.coef_
print(np.count_nonzero(w_4))
```

```
10
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
987
1
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
892
0.1
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
369
0.001
LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=Tr
ue,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
0
```

[5.1.1.1] Calculating sparsity on weight vector obtained using L1 regularization on BOW, SET 1

In [98]:
```
data2= [[0.001,0],[0.1,369],[1,892],[10,987]]
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.draw_sparsity(data2)
```

	Lambda	Non-Zero Columns
1	0.001	0
2	0.1	369
3	1	892
4	10	987

[5.1.2] Applying Logistic Regression with L2 regularization on BOW, SET 1

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

In [81]:
```
logregr.set_lambdaparm(1)
logregr.set_penaltyparm('l2')
print(logregr.getlogRegresion())
logregr.logRegr_fitdata(logregr.xtrain,logregr.ytrain)
w = log_regr.coef_
print(np.count_nonzero(w))
```

```
1
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
1000
```

[5.1.2.1] Performing pertubation test (multicollinearity check) on BOW, SET 1

In [5]:
```
# Please write all the code with proper documentation
print(type(lgrgr_l1.xtrain))
print(lgrgr_l1.xtrain.shape)
```

```
<class 'scipy.sparse.csr.csr_matrix'>
(64000, 1000)
```

In [8]:
```
# clean data
clean_data = lgrgr_l1.xtrain
print(clean_data.shape)
# adding a small epsilon
mu,sigma = 0,0.001
epsilon = np.random.normal(mu,sigma,[64000, 1000])
# data_dash is clean data plus epsilon
data_dash=clean_data+epsilon
```

```
(64000, 1000)
```

In [7]:
```
#instantiate the logistic regression object
lgrgr_pt = LogisticRegrsn()

#instantiate the logistic regression classifier
lgrgrpt_clf = lgrgr_pt.logRegrsn()
print(lgrgrpt_clf)

# set the x_data as the clean data
lgrgr_pt.xtrain = lgrgr_l1.xtrain
lgrgr_pt.ytrain = lgrgr_l1.ytrain

# set lambda equals to 1
lgrgr_pt.set_lambdaparm(1)

# set l2 regularization
lgrgr_pt.set_penaltyparm('l2')

#fit the clean data
lgrgr_pt.logRegr_fitdata(lgrgr_pt.xtrain,lgrgr_pt.ytrain)

# store the coefficients as weights into the weight clean vector
w_clean = lgrgrpt_clf.coef_
print(np.count_nonzero(w_clean))
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
1
```

```
E:\anaconda352\envs\AmaazonFoodReview\lib\site-packages\sklearn\linear_mode
l\logistic.py:433: FutureWarning: Default solver will be changed to 'lbfgs'
in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

```
1000
```

In [10]:
```
#store data_dash (clean+epsilon) into training data
lgrgr_pt.xtrain = data_dash

#fit this new training data
lgrgr_pt.logRegr_fitdata(lgrgr_pt.xtrain,lgrgr_pt.ytrain)

#store the coefficients as weights to w_dash vector
w_dash = lgrgrpt_clf.coef_
print(np.count_nonzero(w_dash))
```

```
1000
```

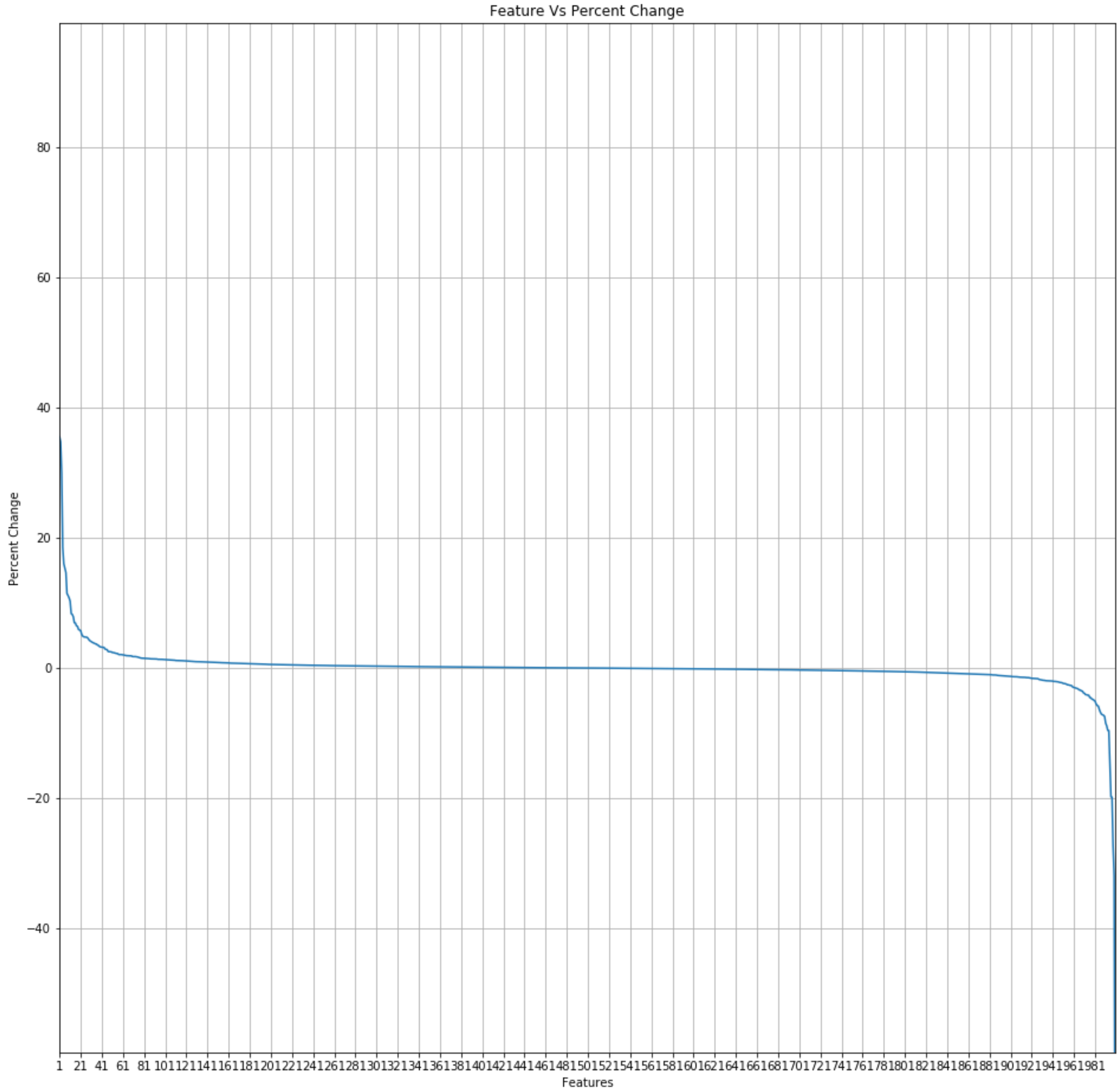
In [11]:
```
print(type(w_clean),w_clean.shape)
print(type(w_dash),w_dash.shape)
```

```
<class 'numpy.ndarray'> (1, 1000)
<class 'numpy.ndarray'> (1, 1000)
```

In [12]:
```
#finding the percentage change in weights
w_tmp = np.empty([1,1000])
w_tmp1 = np.empty([1,1000])
w_clean = w_clean + 10**-6
w_dash = w_dash + 10**-6
np.subtract(w_clean,w_dash,out=w_tmp)
np.divide(w_tmp,w_clean,out=w_tmp1)
w_perchng = [wi*100 for wi in w_tmp1]
```

In the w_perchange vector we need to find the point where the percentage drastically changes. First we plot all the data and get an initial estimate of the number of features where the data changes. This is depicted in Figure 1. Around x=200 the graph is almost near zero. In figure 2 we are plotting points till x=200 and we can see that around x=130 the change in values are constant. Finally in Figure 3. after x=76 the values almost remain constant. The conclusion is we can drop these 76 features.

In [16]:
```
w_tmpsor = -np.sort(-np.asarray(w_perchng))
w_sortedperchng = w_tmpsor[::-1]
w_1 = w_sortedperchng[:,:1000]
data_x = []
for x in range(1,1001):
    data_x.append(x)

x1=np.asarray(data_x)
x1 = x1.reshape(-1,1)

y1=np.asarray(w_1)
y1 = y1.reshape(-1,1)

plt.figure(figsize=(16,16))
plt.axis([1, 1000,-59,99])
plt.plot(x1,y1, label='Percent Change')

plt.xticks(np.arange(min(x1), 1000+1,20 ))
plt.xlabel('Features')
plt.ylabel('Percent Change')
plt.title('Fig: 1 Feature Vs Percent Change')
plt.grid(True)
#plt.legend()
plt.show()
```


Feature Vs Percent Change
80
60
40
20
0
−20
−40
Percent Change
Features

In [29]:
```
w_tmpsor = -np.sort(-np.asarray(w_perchng))
w_sortedperchng = w_tmpsor[::-1]
w_1 = w_sortedperchng[:,:200]
data_x = []
for x in range(1,201):
    data_x.append(x)

x1=np.asarray(data_x)
x1 = x1.reshape(-1,1)

y1=np.asarray(w_1)
y1 = y1.reshape(-1,1)

plt.figure(figsize=(16,16))
plt.axis([1, 200,0,40])
plt.plot(x1,y1, label='Percent Change')

plt.xticks(np.arange(min(x1), 200+1,5.0 ))
plt.xlabel('Features')
plt.ylabel('Percent Change')
plt.title('Fig: 2 Feature Vs Percent Change')
plt.grid(True)
#plt.legend()
plt.show()
```

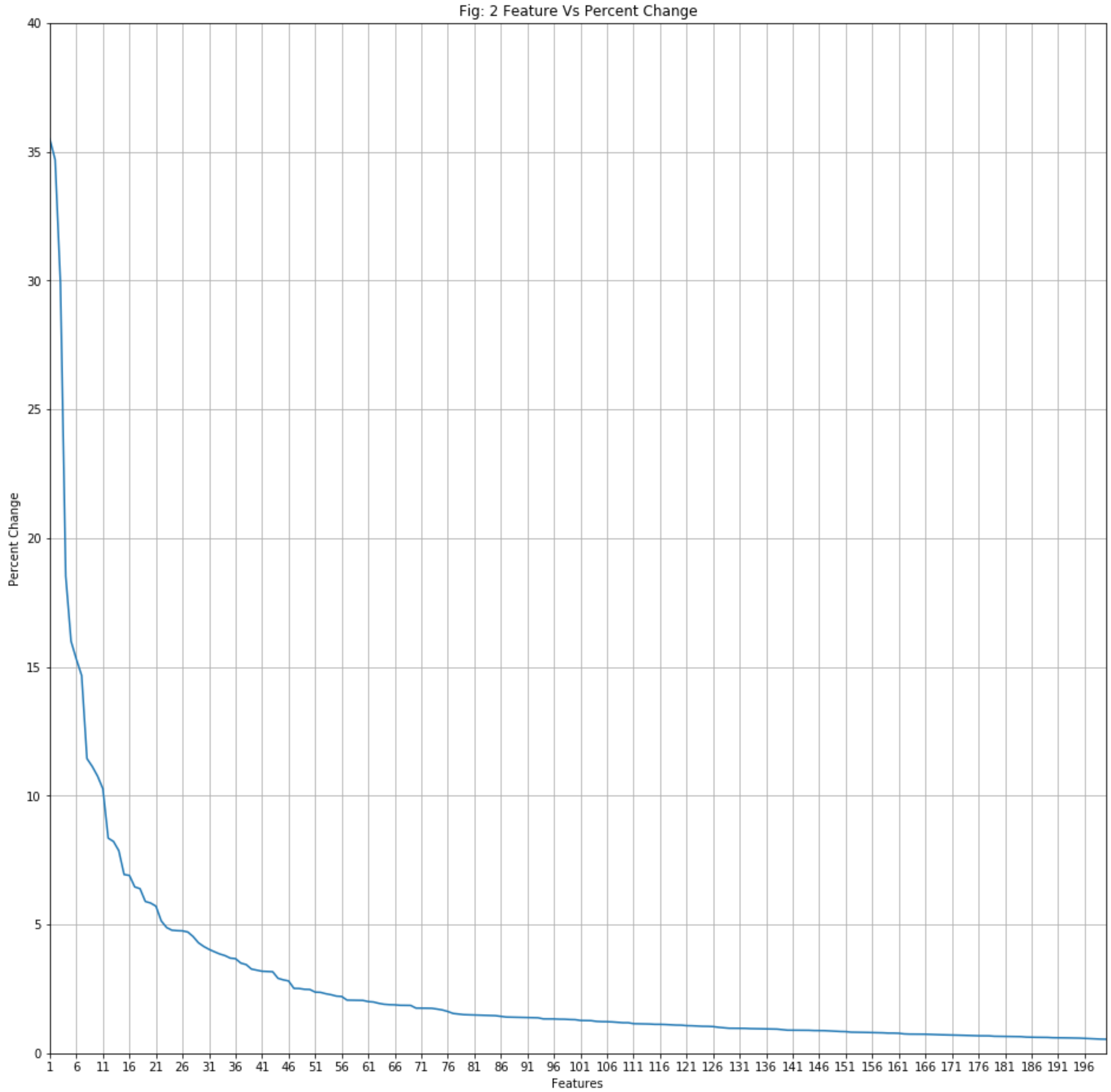


Fig: 2 Feature Vs Percent Change
Percent Change
Features
0
5
10
15
20
25
30
35
40
1 6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96 101 106 111 116 121 126 131 136 141 146 151 156 161 166 171 176 181 186 191 196

In [28]:
```
w_tmpsor = -np.sort(-np.asarray(w_perchng))
w_sortedperchng = w_tmpsor[::-1]
w_1 = w_sortedperchng[:,:130]
data_x = []
for x in range(1,131):
    data_x.append(x)

x1=np.asarray(data_x)
x1 = x1.reshape(-1,1)

y1=np.asarray(w_1)
y1 = y1.reshape(-1,1)

plt.figure(figsize=(16,16))
plt.axis([1, 130,0,40])
plt.plot(x1,y1, label='Percent Change')

plt.xticks(np.arange(min(x1), 130+1,5.0 ))
plt.xlabel('Features')
plt.ylabel('Percent Change')
plt.title('Fig: 3 Feature Vs Percent Change')
plt.grid(True)
#plt.legend()
plt.show()
```

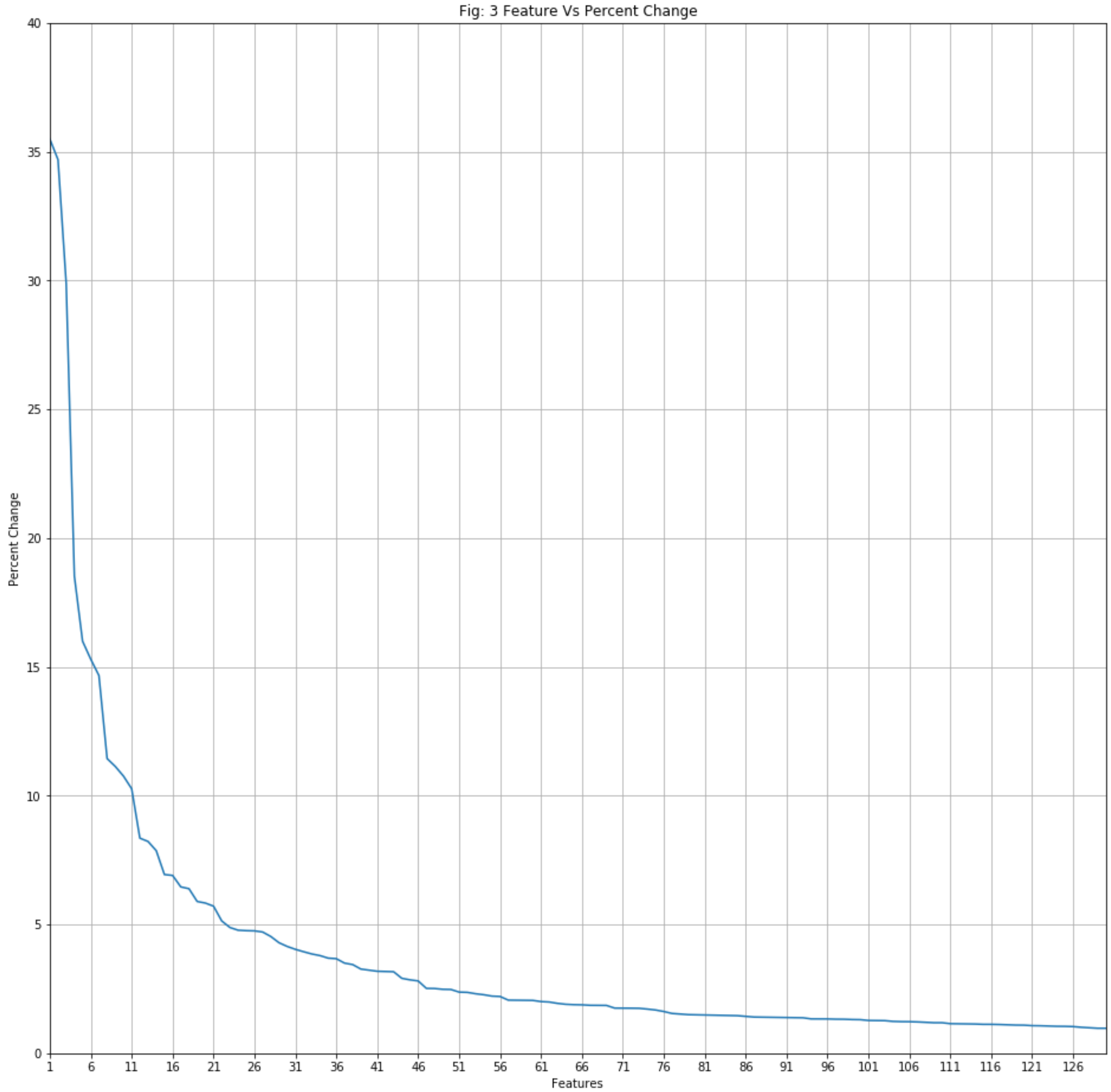


Fig: 3 Feature Vs Percent Change
Percent Change
Features
0
5
10
15
20
25
30
35
40
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
76
81
86
91
96
101
106
111
116
121
126

In [85]:
```
from scipy import stats
# pre-sort array
w_tmp_1 = np.asarray(w_perchng)

w_tmpsor_1 = (w_tmp_1).ravel()
w_tmpsor_1 = sorted(np.asarray(w_tmpsor_1),key=float)
print(type(w_tmpsor_1))
#print(w_tmpsor_1)

# calculate percentiles using scipy func percentileofscore on each array el
ement
df_perchng = pd.Series(w_tmpsor_1)
prcntls = df_perchng.apply(lambda x: stats.percentileofscore(w_tmpsor_1, x
))

#checking that the results are correct:
df = pd.DataFrame({'data': df_perchng, 'percentiles': prcntls, 'featurename
s': lgrgr_l1.count_vect.get_feature_names()})
print(df)
#df_sorted = df.sort_values(by='data')
#print(df_sorted)
```

```
<class 'list'>
          data  percentiles featurenames
0   -83.690190          0.1          abl
1   -31.915668          0.2      absolut
2   -27.138467          0.3         acid
3   -19.825809          0.4       across
4   -19.774519          0.5       actual
5   -14.920021          0.6           ad
6    -9.606266          0.7          add
7    -9.573110          0.8       addict
8    -8.890652          0.9        addit
9    -8.458384          1.0     advertis
10   -7.447325          1.1    aftertast
11   -7.258504          1.2        again
12   -7.186293          1.3          age
13   -7.083141          1.4          ago
14   -6.791987          1.5         agre
15   -6.366837          1.6          air
16   -5.832302          1.7          all
17   -5.779219          1.8       allerg
18   -5.533660          1.9      allergi
19   -5.099714          2.0        allow
20   -4.964140          2.1       almond
21   -4.824241          2.2       almost
22   -4.794247          2.3         alon
23   -4.560981          2.4        along
24   -4.547341          2.5      alreadi
25   -4.183919          2.6         also
26   -4.175691          2.7       altern
27   -4.106289          2.8     although
28   -4.097463          2.9        alway
29   -3.899542          3.0           am
..         ...          ...          ...
970   4.146755         97.1        which
971   4.290953         97.2        while
972   4.532594         97.3        white
973   4.707718         97.4        whole
974   4.752071         97.5         wife
975   4.760027         97.6         will
976   4.777793         97.7         wish
977   4.883024         97.8         with
978   5.137780         97.9       within
979   5.710770         98.0      without
980   5.832323         98.1       wonder
981   5.893502         98.2         word
982   6.389208         98.3         work
983   6.459861         98.4        world
984   6.904483         98.5        worri
985   6.939087         98.6        worth
986   7.865915         98.7        would
987   8.223354         98.8          wow
988   8.352489         98.9         wrap
989  10.269442         99.0        write
990  10.760528         99.1        wrong
991  11.135567         99.2           ye
992  11.440977         99.3         year
993  14.666448         99.4        yeast
```

```
994  15.300868         99.5       yellow
995  15.999656         99.6          yet
996  18.523163         99.7       yogurt
997  29.895689         99.8          you
998  34.698758         99.9          yum
999  35.499030        100.0        yummi

[1000 rows x 3 columns]
```

In [63]:
```
uselesfeat = df.where(df['data']>29.895689).dropna()
#uselesfeat.dropna()
print(uselesfeat)
lgrgr_l1.count_vect.get_feature_names()[0:15]
```

```
          data  percentiles featurenames
15   34.698758         99.9          air
558  35.499030        100.0        night
```

Out[63]:
```
['abl',
 'absolut',
 'acid',
 'across',
 'actual',
 'ad',
 'add',
 'addict',
 'addit',
 'advertis',
 'aftertast',
 'again',
 'age',
 'ago',
 'agre']
```

[5.1.3] Feature Importance on BOW, SET 1

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

In [196]:
```
# Please write all the code with proper documentation
```

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

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

In [61]:
```
feat1_pos=[]
feat0_neg=[]
features1=[]

class_labels = log_regr.classes_
feature_names = logregr.count_vect.get_feature_names()
top10n_neg = sorted(zip((log_regr.predict_proba(logregr.xtest))[:,0], featu
re_names),reverse=True)[:10]
top10n_pos = sorted(zip((log_regr.predict_proba(logregr.xtest))[:,1], featu
re_names),reverse=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	last	brought
2	again	diet
3	trip	certainli
4	slice	super
5	shipment	tomato
6	stock	servic
7	browni	see
8	bag	piec
9	jelli	some
10	never	crumbl

[5.2] Logistic Regression on TFIDF, SET 2

In [62]:
```
#instantiate logistic regression object and classifier
lgrgr_tfidf = LogisticRegrsn()
lgrgrtfidf_clf = lgrgr_tfidf.logRegrsn()

# load the data
lgrgr_tfidf.xtrain,lgrgr_tfidf.xtest,lgrgr_tfidf.xval, lgrgr_tfidf.ytrain,l
grgr_tfidf.ytest,lgrgr_tfidf.yval = lgrgr_tfidf.load_data()

# vectorise the complete corpus
lgrgr_tfidf.tfIdfVectorizer()

# print the shapes of the data vetors
print((lgrgr_tfidf.xtrain).shape)
print((lgrgr_tfidf.xtest).shape)
print((lgrgr_tfidf.xval).shape)
print((lgrgr_tfidf.ytrain).shape)
print((lgrgr_tfidf.ytest).shape)
print((lgrgr_tfidf.yval).shape)
```

```
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, 40359)
the number of unique words including both unigrams and bigrams  40359
(64000, 40359)
(20000, 40359)
(16000, 40359)
(64000,)
(20000,)
(16000,)
```

In [63]:
```
print(lgrgrtfidf_clf)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

In [64]:
```
# hyperparameter tuning for lambda
print(lgrgr_tfidf.getlogRegresion())
return_63 = lgrgr_tfidf.hyperparamtuning(wordvect.TFIDF,[0.00000000001,0.00
00000001,0.0000000001,0.000000001,0.00000001,0.0000001,0.000001,0.00001,0.0
001,0.001,0.01,1,10,100,1000,10000,100000,1000000,10000000,100000000,100000
0000,10000000000],'roc_auc',5,100,1)

print(return_63[0])
print(return_63[1])
print(return_63[2])
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
Fitting 5 folds for each of 22 candidates, totalling 110 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.6146325444184904, total=   0.1s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.6287481174746223, total=   0.1s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.3s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.6303265016424122, total=   0.1s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.6s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.6157025626765039, total=   0.1s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.8s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.6169602814326911, total=   0.1s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    1.0s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.6146325444184904, total=   0.1s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    1.2s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.6287481174746223, total=   0.1s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    1.4s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.6303265016424122, total=   0.1s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    1.6s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.6157025626765039, total=   0.1s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.9s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.6169602814326911, total=   0.1s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    2.1s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.6146325444184904, total=   0.1s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    2.3s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.6287481174746223, total=   0.1s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    2.5s remaining:
0.0s
[CV] C=1e-10 .........................................................
```

```
[CV] ................ C=1e-10, score=0.6303265016424122, total=   0.1s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    2.7s remaining:
0.0s
[CV] C=1e-10 ...........................................................
[CV] ................ C=1e-10, score=0.6157025626765039, total=   0.1s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    2.9s remaining:
0.0s
[CV] C=1e-10 ...........................................................
[CV] ................ C=1e-10, score=0.6169602814326911, total=   0.1s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:    3.2s remaining:
0.0s
[CV] C=1e-09 ...........................................................
[CV] ................ C=1e-09, score=0.6146325444184904, total=   0.1s
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed:    3.4s remaining:
0.0s
[CV] C=1e-09 ...........................................................
[CV] ................ C=1e-09, score=0.6287481174746223, total=   0.1s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed:    3.6s remaining:
0.0s
[CV] C=1e-09 ...........................................................
[CV] ................ C=1e-09, score=0.6303265016424122, total=   0.1s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    3.8s remaining:
0.0s
[CV] C=1e-09 ...........................................................
[CV] ................ C=1e-09, score=0.6157025626765039, total=   0.1s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed:    4.0s remaining:
0.0s
[CV] C=1e-09 ...........................................................
[CV] ................ C=1e-09, score=0.6169602814326911, total=   0.1s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    4.2s remaining:
0.0s
[CV] C=1e-08 ...........................................................
[CV] ................ C=1e-08, score=0.6146325444184904, total=   0.1s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    4.4s remaining:
0.0s
[CV] C=1e-08 ...........................................................
[CV] ................ C=1e-08, score=0.6287481174746223, total=   0.1s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed:    4.7s remaining:
0.0s
[CV] C=1e-08 ...........................................................
[CV] ................ C=1e-08, score=0.6303265016424122, total=   0.1s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:    4.9s remaining:
0.0s
[CV] C=1e-08 ...........................................................
[CV] ................ C=1e-08, score=0.6157025626765039, total=   0.1s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:    5.1s remaining:
0.0s
[CV] C=1e-08 ...........................................................
[CV] ................ C=1e-08, score=0.6169602814326911, total=   0.1s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    5.3s remaining:
0.0s
[CV] C=1e-07 ...........................................................
[CV] ................ C=1e-07, score=0.6147521675083727, total=   0.1s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    5.6s remaining:
0.0s
[CV] C=1e-07 ...........................................................
[CV] ................ C=1e-07, score=0.6288702115153285, total=   0.1s
```

```
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    5.8s remaining:
0.0s
[CV] C=1e-07 ..........................................................
[CV] ................. C=1e-07, score=0.630452289414988, total=   0.1s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    6.1s remaining:
0.0s
[CV] C=1e-07 ..........................................................
[CV] ................ C=1e-07, score=0.6158371918630255, total=   0.1s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    6.4s remaining:
0.0s
[CV] C=1e-07 ..........................................................
[CV] ................ C=1e-07, score=0.6170869114057537, total=   0.1s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    6.7s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ................ C=1e-06, score=0.6158545991759233, total=   0.1s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    6.9s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ................ C=1e-06, score=0.6299958991906327, total=   0.2s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    7.3s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ................ C=1e-06, score=0.6315755606568408, total=   0.2s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    7.6s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ................ C=1e-06, score=0.6170202997736756, total=   0.1s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    7.8s remaining:
0.0s
[CV] C=1e-06 ..........................................................
[CV] ................ C=1e-06, score=0.6182116977782742, total=   0.1s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    8.1s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ................ C=1e-05, score=0.6268197583817073, total=   0.2s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    8.4s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ................ C=1e-05, score=0.6410701678327939, total=   0.2s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:    8.7s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ................ C=1e-05, score=0.6426265204039784, total=   0.2s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:    9.1s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ................ C=1e-05, score=0.6287094292013372, total=   0.2s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:    9.4s remaining:
0.0s
[CV] C=1e-05 ..........................................................
[CV] ................ C=1e-05, score=0.6293372947377587, total=   0.2s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    9.7s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ............... C=0.0001, score=0.7175968229967808, total=   0.2s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:   10.1s remaining:
```

```
0.0s
[CV] C=0.0001 ........................................................
[CV] ............... C=0.0001, score=0.7317369602111026, total=   0.2s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:   10.4s remaining:
0.0s
[CV] C=0.0001 ........................................................
[CV] ............... C=0.0001, score=0.7329391369369496, total=   0.2s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:   10.8s remaining:
0.0s
[CV] C=0.0001 ........................................................
[CV] ............... C=0.0001, score=0.7244858057108373, total=   0.2s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:   11.1s remaining:
0.0s
[CV] C=0.0001 ........................................................
[CV] ............... C=0.0001, score=0.7207389741265683, total=   0.3s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:   11.6s remaining:
0.0s
[CV] C=0.001 .........................................................
[CV] ................ C=0.001, score=0.9016770876191411, total=   0.3s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:   12.0s remaining:
0.0s
[CV] C=0.001 .........................................................
[CV] ................ C=0.001, score=0.9011155519319249, total=   0.3s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:   12.4s remaining:
0.0s
[CV] C=0.001 .........................................................
[CV] ................ C=0.001, score=0.9124260294246479, total=   0.3s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:   12.8s remaining:
0.0s
[CV] C=0.001 .........................................................
[CV] ................ C=0.001, score=0.9082761704691534, total=   0.3s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:   13.2s remaining:
0.0s
[CV] C=0.001 .........................................................
[CV] ................ C=0.001, score=0.9003958786500896, total=   0.3s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:   13.7s remaining:
0.0s
[CV] C=0.01 ..........................................................
[CV] ................. C=0.01, score=0.9150162976164335, total=   0.4s
[Parallel(n_jobs=1)]: Done  51 out of  51 | elapsed:   14.3s remaining:
0.0s
[CV] C=0.01 ..........................................................
[CV] ................. C=0.01, score=0.9118133154985595, total=   0.5s
[Parallel(n_jobs=1)]: Done  52 out of  52 | elapsed:   14.9s remaining:
0.0s
[CV] C=0.01 ..........................................................
[CV] ................. C=0.01, score=0.9238704393695183, total=   0.4s
[Parallel(n_jobs=1)]: Done  53 out of  53 | elapsed:   15.4s remaining:
0.0s
[CV] C=0.01 ..........................................................
[CV] ................. C=0.01, score=0.9226897835209221, total=   0.4s
[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:   16.0s remaining:
0.0s
[CV] C=0.01 ..........................................................
[CV] ................. C=0.01, score=0.9143279875316987, total=   0.4s
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:   16.6s remaining:
0.0s
```

```
[CV] C=1 ............................................................
[CV] .................... C=1, score=0.9551713803041382, total=   1.2s
[Parallel(n_jobs=1)]: Done  56 out of  56 | elapsed:   17.8s remaining:
0.0s
[CV] C=1 ............................................................
[CV] .................... C=1, score=0.9560351471921181, total=   1.1s
[Parallel(n_jobs=1)]: Done  57 out of  57 | elapsed:   19.1s remaining:
0.0s
[CV] C=1 ............................................................
[CV] .................... C=1, score=0.9619710033018805, total=   1.2s
[Parallel(n_jobs=1)]: Done  58 out of  58 | elapsed:   20.4s remaining:
0.0s
[CV] C=1 ............................................................
[CV] ..................... C=1, score=0.956505173697345, total=   1.2s
[Parallel(n_jobs=1)]: Done  59 out of  59 | elapsed:   21.7s remaining:
0.0s
[CV] C=1 ............................................................
[CV] .................... C=1, score=0.9552699012189249, total=   1.2s
[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:   22.9s remaining:
0.0s
[CV] C=10 ...........................................................
[CV] ................... C=10, score=0.9572845277586616, total=   1.9s
[Parallel(n_jobs=1)]: Done  61 out of  61 | elapsed:   25.0s remaining:
0.0s
[CV] C=10 ...........................................................
[CV] ................... C=10, score=0.9581107458841226, total=   1.9s
[Parallel(n_jobs=1)]: Done  62 out of  62 | elapsed:   27.1s remaining:
0.0s
[CV] C=10 ...........................................................
[CV] ................... C=10, score=0.9645043477215939, total=   1.9s
[Parallel(n_jobs=1)]: Done  63 out of  63 | elapsed:   29.1s remaining:
0.0s
[CV] C=10 ...........................................................
[CV] ................... C=10, score=0.9576333536755756, total=   2.0s
[Parallel(n_jobs=1)]: Done  64 out of  64 | elapsed:   31.3s remaining:
0.0s
[CV] C=10 ...........................................................
[CV] .................... C=10, score=0.957782914727221, total=   2.0s
[Parallel(n_jobs=1)]: Done  65 out of  65 | elapsed:   33.4s remaining:
0.0s
[CV] C=100 ..........................................................
[CV] .................. C=100, score=0.9509955218619071, total=   2.5s
[Parallel(n_jobs=1)]: Done  66 out of  66 | elapsed:   36.0s remaining:
0.0s
[CV] C=100 ..........................................................
[CV] .................. C=100, score=0.9517908288770623, total=   3.1s
[Parallel(n_jobs=1)]: Done  67 out of  67 | elapsed:   39.3s remaining:
0.0s
[CV] C=100 ..........................................................
[CV] .................. C=100, score=0.9591551209384503, total=   2.9s
[Parallel(n_jobs=1)]: Done  68 out of  68 | elapsed:   42.3s remaining:
0.0s
[CV] C=100 ..........................................................
[CV] .................. C=100, score=0.9516328770194136, total=   2.8s
[Parallel(n_jobs=1)]: Done  69 out of  69 | elapsed:   45.2s remaining:
0.0s
[CV] C=100 ..........................................................
```

```
[CV] .................. C=100, score=0.9511143825470155, total=   2.8s
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:   48.1s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9471848796414388, total=   3.1s
[Parallel(n_jobs=1)]: Done  71 out of  71 | elapsed:   51.3s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9479531515475808, total=   3.2s
[Parallel(n_jobs=1)]: Done  72 out of  72 | elapsed:   54.7s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9557334124885358, total=   3.3s
[Parallel(n_jobs=1)]: Done  73 out of  73 | elapsed:   58.1s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9481904518673268, total=   3.6s
[Parallel(n_jobs=1)]: Done  74 out of  74 | elapsed:  1.0min remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.9470360441649665, total=   3.3s
[Parallel(n_jobs=1)]: Done  75 out of  75 | elapsed:  1.1min remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9452990596627067, total=   3.4s
[Parallel(n_jobs=1)]: Done  76 out of  76 | elapsed:  1.1min remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9453658237849656, total=   4.3s
[Parallel(n_jobs=1)]: Done  77 out of  77 | elapsed:  1.2min remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9542110703318393, total=   3.6s
[Parallel(n_jobs=1)]: Done  78 out of  78 | elapsed:  1.3min remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................. C=10000, score=0.946142507781538, total=   3.4s
[Parallel(n_jobs=1)]: Done  79 out of  79 | elapsed:  1.3min remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.9460317792752305, total=   2.8s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:  1.4min remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.9436536486641264, total=   3.5s
[Parallel(n_jobs=1)]: Done  81 out of  81 | elapsed:  1.5min remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.9452940208610265, total=   3.4s
[Parallel(n_jobs=1)]: Done  82 out of  82 | elapsed:  1.5min remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.9538675766801485, total=   3.4s
[Parallel(n_jobs=1)]: Done  83 out of  83 | elapsed:  1.6min remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.9450633411657754, total=   3.2s
```

```
[Parallel(n_jobs=1)]: Done  84 out of  84 | elapsed:  1.6min remaining:
0.0s
[CV] C=100000 .........................................................
[CV] ............... C=100000, score=0.9435405333235597, total=   3.4s
[Parallel(n_jobs=1)]: Done  85 out of  85 | elapsed:  1.7min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9436714782700708, total=   3.4s
[Parallel(n_jobs=1)]: Done  86 out of  86 | elapsed:  1.7min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9434340731909199, total=   3.8s
[Parallel(n_jobs=1)]: Done  87 out of  87 | elapsed:  1.8min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9537812308362619, total=   3.3s
[Parallel(n_jobs=1)]: Done  88 out of  88 | elapsed:  1.9min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] ............... C=1000000, score=0.942857933775045, total=   3.8s
[Parallel(n_jobs=1)]: Done  89 out of  89 | elapsed:  1.9min remaining:
0.0s
[CV] C=1000000 ........................................................
[CV] .............. C=1000000, score=0.9449507219362868, total=   2.8s
[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:  2.0min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9436777283221546, total=   3.4s
[Parallel(n_jobs=1)]: Done  91 out of  91 | elapsed:  2.0min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9451080696990304, total=   3.3s
[Parallel(n_jobs=1)]: Done  92 out of  92 | elapsed:  2.1min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9537666944989072, total=   3.1s
[Parallel(n_jobs=1)]: Done  93 out of  93 | elapsed:  2.2min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9426188300127387, total=   3.8s
[Parallel(n_jobs=1)]: Done  94 out of  94 | elapsed:  2.2min remaining:
0.0s
[CV] C=10000000 .......................................................
[CV] ............. C=10000000, score=0.9452997785235954, total=   2.9s
[Parallel(n_jobs=1)]: Done  95 out of  95 | elapsed:  2.3min remaining:
0.0s
[CV] C=100000000 ......................................................
[CV] ............ C=100000000, score=0.9422539276974594, total=   3.5s
[Parallel(n_jobs=1)]: Done  96 out of  96 | elapsed:  2.3min remaining:
0.0s
[CV] C=100000000 ......................................................
[CV] ............ C=100000000, score=0.9435602854829992, total=   4.0s
[Parallel(n_jobs=1)]: Done  97 out of  97 | elapsed:  2.4min remaining:
0.0s
[CV] C=100000000 ......................................................
[CV] ............ C=100000000, score=0.9537680996781849, total=   3.1s
[Parallel(n_jobs=1)]: Done  98 out of  98 | elapsed:  2.5min remaining:
```

```
0.0s
[CV] C=100000000 .....................................................
[CV] ............ C=100000000, score=0.942754283360647, total=   3.8s
[Parallel(n_jobs=1)]: Done  99 out of  99 | elapsed:  2.5min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ........... C=100000000, score=0.9439380699924391, total=   3.2s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9421376476586916, total=   3.7s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9436030668472626, total=   4.0s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9537628665967371, total=   3.1s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9427941824677797, total=   3.8s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.9453359931445287, total=   2.7s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9436841237242868, total=   3.5s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9436035028974079, total=   4.0s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9537664037721603, total=   3.1s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9427430844618041, total=   3.8s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.9452975969199248, total=   2.9s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:  3.2min finished
0.9590631776253917
{'C': 10}
GridSearchCV(cv=5, error_score='raise-deprecating',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, f
it_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=1,
       param_grid={'C': [1e-11, 1e-10, 1e-10, 1e-09, 1e-08, 1e-07, 1e-06, 1
e-05, 0.0001, 0.001, 0.01, 1, 10, 100, 1000, 10000, 100000, 1000000, 100000
00, 100000000, 1000000000, 10000000000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=100)
```

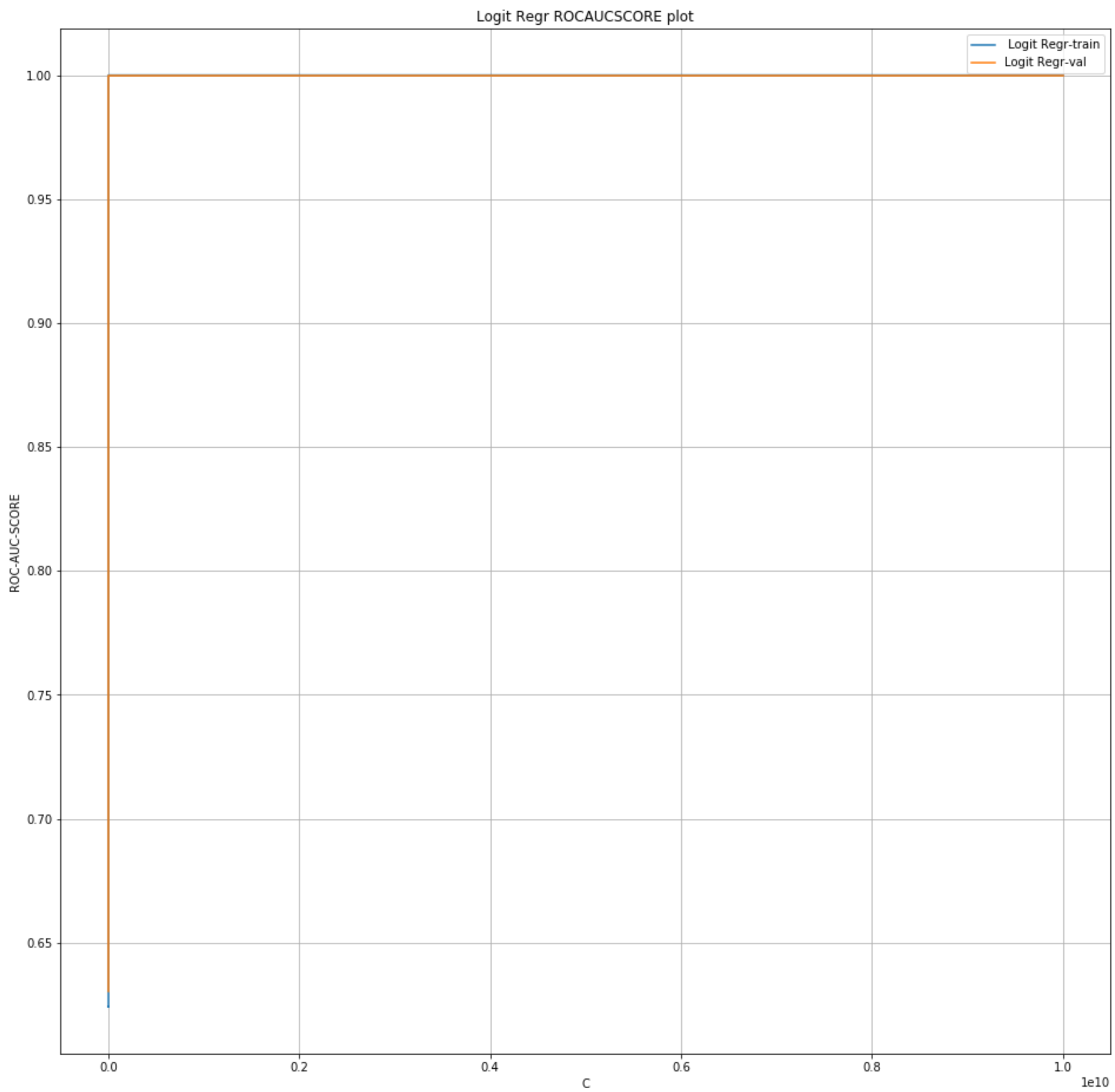
In [65]:
```
#run with different lambdas to get multiple rocauc scores
lgrgr_tfidf.calcrocaucscore_logregrsn(10000000000)
print(lgrgr_tfidf.rocaucscoretrn)
print(lgrgr_tfidf.rocaucscoreval)
print( lgrgr_tfidf.logrgr_lambda)
```

```
1e-11
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...
9.999999999999999e-11
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...
9.999999999999999e-10
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...
9.999999999999999e-09
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...
9.999999999999998e-08
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...
9.999999999999997e-07
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...
9.999999999999997e-06
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...
9.999999999999998e-05
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...
0.0009999999999999998
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...
0.009999999999999998
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...
0.09999999999999998
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...
0.9999999999999998
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...
9.999999999999998
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...
99.99999999999999
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...
999.9999999999999
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...
9999.999999999998
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...
99999.99999999999
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...
999999.9999999999
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...
9999999.999999998
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...
```

```
99999999.99999999
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...
999999999.9999999
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...
9999999999.999998
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...
[0.6240641808808883, 0.6240641905734738, 0.6240641876656982, 0.624064188634
9567, 0.6242259733932017, 0.6256831062788344, 0.6399511463843335, 0.7505725
090332764, 0.9153740243860331, 0.9253578913490488, 0.9501521799876818, 0.98
00974768191935, 0.9986111077617843, 0.9999977038265219, 0.9999998788426826,
0.9999999893381561, 0.9999999893381561, 0.999999989338156, 0.99999998933815
6, 0.9999999893381561, 0.999999989338156, 0.9999999893381561]
[0.6302816120395272, 0.6302815196196342, 0.6302815196196342, 0.630281519619
6342, 0.6302815196196342, 0.6307300025532535, 0.6346872683341351, 0.6722241
315487504, 0.862046181357897, 0.9357341022997028, 0.9488529520638965, 0.985
6310171511606, 0.9999652809269001, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
1.0]
[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, 999999999.9999999, 9999999999.999998]
```

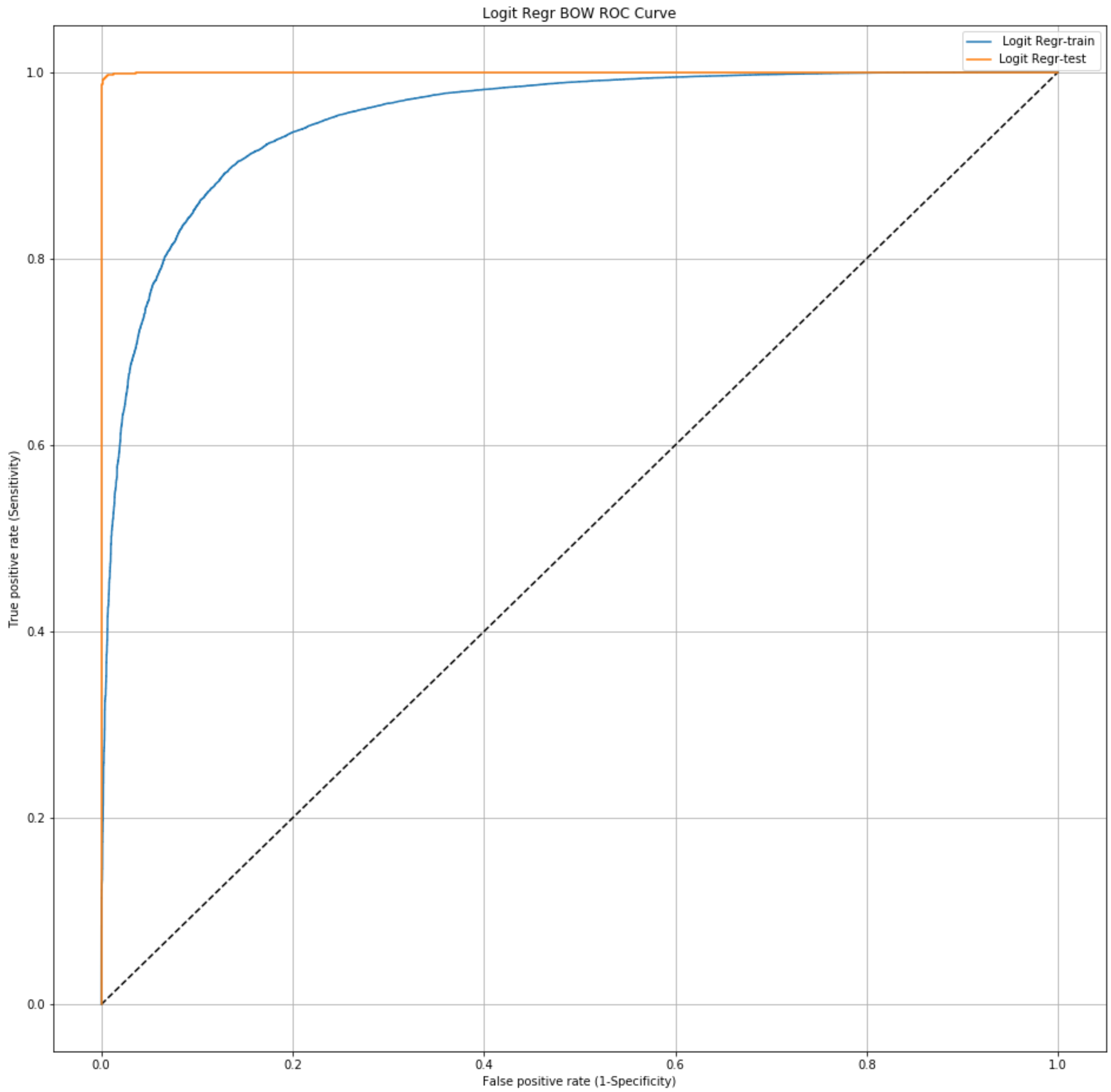
In [66]:
```
#display rocauc scores
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr ROCAUCSCORE plot'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-val'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='C'
displaygraph.label_y='ROC-AUC-SCORE'
displaygraph.Xdata = lgrgr_tfidf.logrgr_lambda
displaygraph.ydatatrn = lgrgr_tfidf.rocaucscoretrn
displaygraph.ydataval = lgrgr_tfidf.rocaucscoreval
displaygraph.rocacuscoregraph()
```



In [67]:
```
#output of gridsearchcv lambda equals 10 use that with the teest data
#process logistic regression
lgrgr_tfidf.actualClasifier_logregrsn(10)
```

```
10
```

In [68]:
```
#display the output of logistic regression of test data
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr BOW ROC Curve'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-test'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='False positive rate (1-Specificity)'
displaygraph.label_y='True positive rate (Sensitivity)'
displaygraph.constructgraph(lgrgr_tfidf.roc_curve_test['fpr_trn'],lgrgr_tfidf.roc_curve_test['tpr_trn'],\
                            lgrgr_tfidf.roc_curve_test['fpr'],lgrgr_tfidf.roc_curve_test['tpr'])
data = [[lgrgr_tfidf.confsnmtxytstpred['tn'] ,lgrgr_tfidf.confsnmtxytstpred['fn']],[lgrgr_tfidf.confsnmtxytstpred['fp'],lgrgr_tfidf.confsnmtxytstpred['tp']]]
displaygraph.draw_table(data)
```



	Predicted: NO	Predicted: YES
Actual: NO	2854	17
Actual: YES	107	17022

In [69]:
```
data1= [[lgrgr_tfidf.accuracy_score_val,lgrgr_tfidf.accuracy_score_test]]
#data1=[[0,1]]
displaygraph.draw_accscore(data1)
```

	Validation	Test
Accuracy Score	1.0	0.9938

[5.2.1] Applying Logistic Regression with L1 regularization on TFIDF, SET 2

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

In [60]:
```
# instantiate logistic regression object
lgrgr_tfidf_l1 = LogisticRegrsn()
#instantiate logistic regression clasifier
lgrgrtfidf_clf = lgrgr_tfidf_l1.logRegrsn()

#load the data into the logistic regression object
lgrgr_tfidf_l1.xtrain,lgrgr_tfidf_l1.xtest,lgrgr_tfidf_l1.xval, lgrgr_tfidf
_l1.ytrain,lgrgr_tfidf_l1.ytest,lgrgr_tfidf_l1.yval = lgrgr_tfidf_l1.load_d
ata()

# vectorise the training corpus
lgrgr_tfidf_l1.tfIdfVectorizer()

# print the shapes of the data vetors
print((lgrgr_tfidf_l1.xtrain).shape)
print((lgrgr_tfidf_l1.xtest).shape)
print((lgrgr_tfidf_l1.xval).shape)
print((lgrgr_tfidf_l1.ytrain).shape)
print((lgrgr_tfidf_l1.ytest).shape)
print((lgrgr_tfidf_l1.yval).shape)
```

```
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, 40359)
the number of unique words including both unigrams and bigrams  40359
(64000, 40359)
(20000, 40359)
(16000, 40359)
(64000,)
(20000,)
(16000,)
```

In [61]:
```
# set arbitary values of lambda
lgrgr_tfidf_l1.set_lambdaparm(10)

# set penalty equals l1
lgrgr_tfidf_l1.set_penaltyparm('l1')

#print the logistic regression classifier to check the parameters
print(lgrgr_tfidf_l1.getlogRegresion())

# fir the test data
lgrgr_tfidf_l1.logRegr_fitdata(lgrgr_tfidf_l1.xtest,lgrgr_tfidf_l1.ytest)

#get the handle to the logistic regression classifier
lone_tst_1= lgrgr_tfidf_l1.getlogRegresion()

#get the coefficients
w_1 = lone_tst_1.coef_
print(np.count_nonzero(w_1))

# set arbitary values of lambda
lgrgr_tfidf_l1.set_lambdaparm(1)

# set penalty equals l1
lgrgr_tfidf_l1.set_penaltyparm('l1')

#print the logistic regression classifier to check the parameters
print(lgrgr_tfidf_l1.getlogRegresion())

# fir the test data
lgrgr_tfidf_l1.logRegr_fitdata(lgrgr_tfidf_l1.xtest,lgrgr_tfidf_l1.ytest)

#get the coefficients
lone_tst_2 = lgrgr_tfidf_l1.getlogRegresion()
w_2 = lone_tst_2.coef_
print(np.count_nonzero(w_2))

# set arbitary values of lambda
lgrgr_tfidf_l1.set_lambdaparm(0.1)

# set penalty equals l1
lgrgr_tfidf_l1.set_penaltyparm('l1')

#print the logistic regression classifier to check the parameters
print(lgrgr_tfidf_l1.getlogRegresion())

# fir the test data
lgrgr_tfidf_l1.logRegr_fitdata(lgrgr_tfidf_l1.xtest,lgrgr_tfidf_l1.ytest)

#get the coefficients
lone_tst_3 = lgrgr_tfidf_l1.getlogRegresion()
w_3 = lone_tst_3.coef_
print(np.count_nonzero(w_3))

# set arbitary values of lambda
lgrgr_tfidf_l1.set_lambdaparm(0.001)
```

```
# set penalty equals l1
lgrgr_tfidf_l1.set_penaltyparm('l1')

#print the logistic regression classifier to check the parameters
print(lgrgr_tfidf_l1.getlogRegresion())

# fir the test data
lgrgr_tfidf_l1.logRegr_fitdata(lgrgr_tfidf_l1.xtest,lgrgr_tfidf_l1.ytest)

#get the coefficients
lone_tst_4 = lgrgr_tfidf_l1.getlogRegresion()
w_4 = lone_tst_4.coef_
print(np.count_nonzero(w_4))
```

```
10
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
3182
1
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
474
0.1
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
28
0.001
LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=Tr
ue,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
0
```

In [4]:
```
data2= [[0.001,0],[0.1,28],[1,474],[10,3182]]
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.draw_sparsity(data2)
```

	Lambda	Non-Zero Columns
1	0.001	0
2	0.1	28
3	1	474
4	10	3182

[5.2.2] Applying Logistic Regression with L2 regularization on TFIDF, SET 2

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

In [71]:
```
#l2 regulariation happends by default use  the previously used logisitc reg
ression classifier
print(lgrgrtfidf_clf)
w = lgrgrtfidf_clf.coef_
print(np.count_nonzero(w))
```

```
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
39209
```

[5.2.3] Feature Importance on TFIDF, SET 2

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

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

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

In [74]:
```
# Please write all the code with proper documentation
```

In [73]:
```
feat1_pos=[]
feat0_neg=[]
features1=[]

class_labels = lgrgrtfidf_clf.classes_
feature_names = lgrgr_tfidf.tf_idf_vect.get_feature_names()
top10n_neg = sorted(zip((lgrgrtfidf_clf.predict_proba(lgrgr_tfidf.xtest))
[:,0], feature_names),reverse=True)[:10]
top10n_pos = sorted(zip((lgrgrtfidf_clf.predict_proba(lgrgr_tfidf.xtest))
[:,1], feature_names),reverse=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	long time	human food
2	arm	come well
3	dad love	issu year
4	dollar	high ship
5	gym bag	littl work
6	like anoth	anyon tri
7	els have	act like
8	feel	like bigelow
9	get refund	box cooki
10	breath deep	is eat

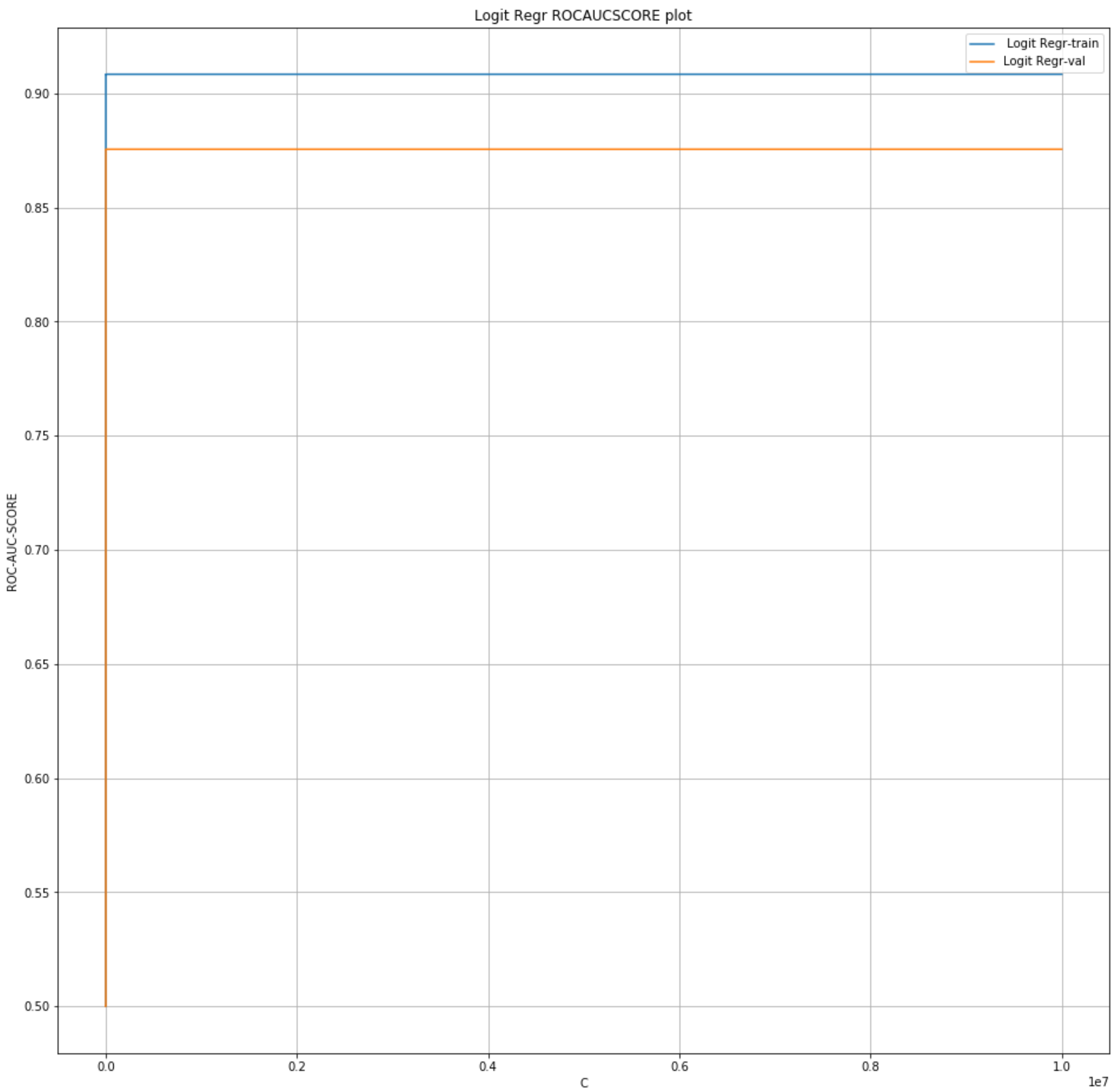
[5.3] Logistic Regression on AVG W2V, SET 3

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

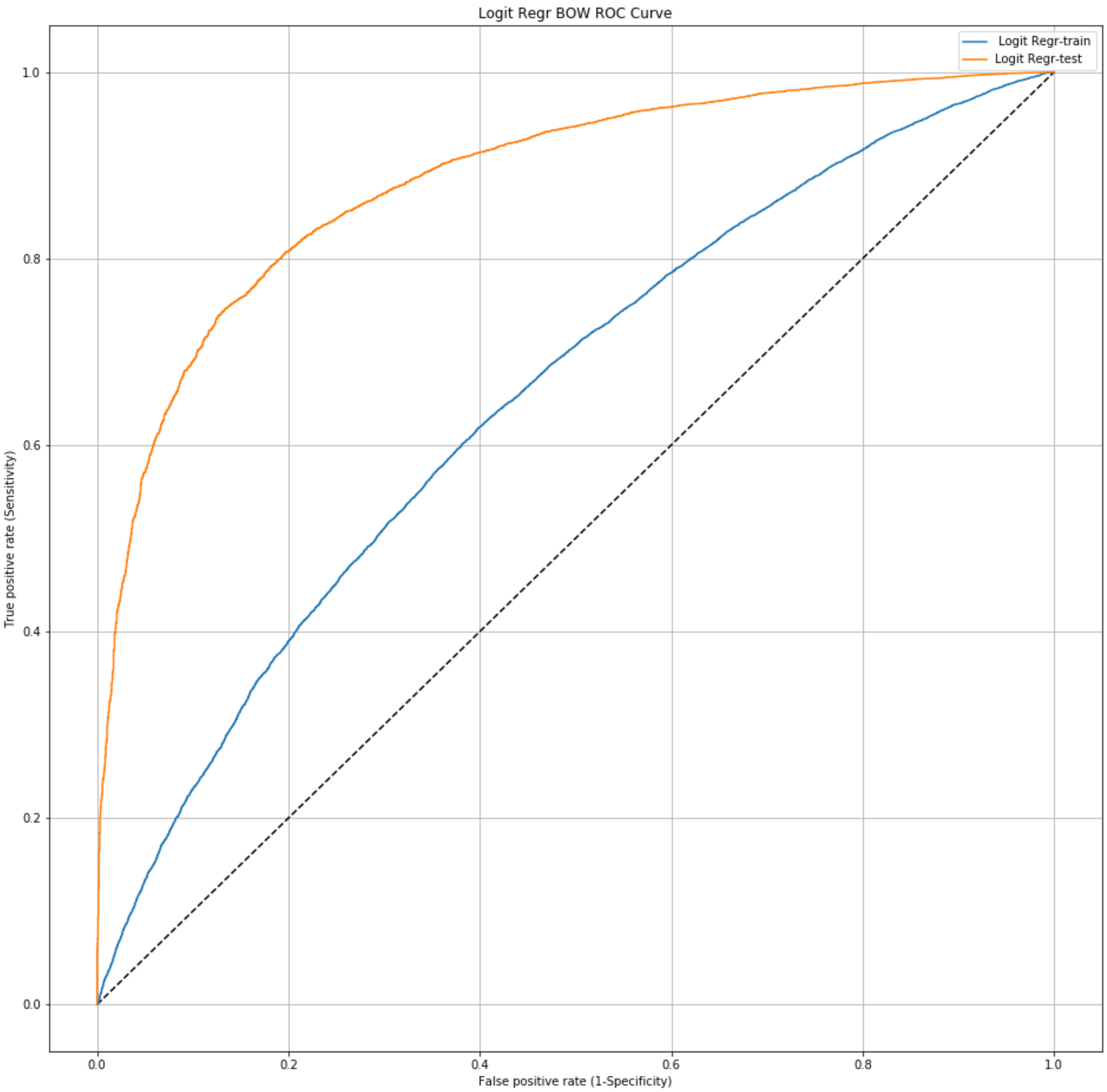
In [12]:
```
# Train your own Word2Vec model using your own text corpus

# initialize the logistic regression object
logregr_avgw2v = LogisticRegrsn()

#initialize logistic regression classifier
log_regr_avgw2v = logregr_avgw2v.logRegrsn()

#load the data
x_train,x_test,x_val, y_trn,y_tst,y_val = logregr_avgw2v.load_data()

# split the data into sentences
listsent_xtrain=[]
listsent_xtest=[]
listsent_xval=[]

listsent_xtrain = logregr_avgw2v.listsent(x_train)
listsent_xtest = logregr_avgw2v.listsent(x_test)
listsent_xval = logregr_avgw2v.listsent(x_val)

#create wordtovec model
w2v_mdl_xtrain=Word2Vec(listsent_xtrain,min_count=5,size=50, workers=4)
w2v_mdl_xtest=Word2Vec(listsent_xtest,min_count=5,size=50, workers=4)
w2v_mdl_xval=Word2Vec(listsent_xval,min_count=5,size=50, workers=4)

#get the vocabulary from the model
w2v_words_trn = list(w2v_mdl_xtrain.wv.vocab)
w2v_words_tst = list(w2v_mdl_xtest.wv.vocab)
w2v_words_val = list(w2v_mdl_xval.wv.vocab)

#create sent vectors for training data
w2v_xtrain = logregr_avgw2v.w2vec_crea(listsent_xtrain,w2v_mdl_xtrain,w2v_w
ords_trn)

#create sent vectors for test data
w2v_xtest = logregr_avgw2v.w2vec_crea(listsent_xtest,w2v_mdl_xtest,w2v_word
s_tst)

#create sent vectors for validation data
w2v_xval = logregr_avgw2v.w2vec_crea(listsent_xval,w2v_mdl_xval,w2v_words_v
al)

logregr_avgw2v.xtrain = w2v_xtrain
logregr_avgw2v.xtest = w2v_xtest
logregr_avgw2v.xval = w2v_xval
logregr_avgw2v.ytrain = y_trn
logregr_avgw2v.ytest = y_tst
logregr_avgw2v.yval = y_val

print(len(logregr_avgw2v.xtrain)
print(len(logregr_avgw2v.xtest)
print(len(logregr_avgw2v.xval)

#parameter tuning lambda
return_hyparmtune = logregr_avgw2v.hyperparamtuning(wordvect.W2VAVG,[0.0000
0000001,0.0000000001,0.0000000001,0.000000001,0.00000001,0.0000001,0.000001
```

```
,0.00001,0.0001,0.001,0.01,1,10,100,1000,10000,100000,1000000,10000000,1000
00000,1000000000,10000000000],'roc_auc',5,100,1)

print(return_hyparmtune[0],return_hyparmtune[1],return_hyparmtune[2])
#output hyperparmtuning 0.9053700080793041 {'C': 1}

#Process the training and validation data sets using logistic regression
#calculate roc_auc_score
logregr_avgw2v.calcrocaucscore_logregrsn(10000000)

print(logregr_avgw2v.rocaucscoretrn)
print(logregr_avgw2v.rocaucscoreval)
print( logregr_avgw2v.logrgr_lambda)


# plot training and validation datasets roc_auc score
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr ROCAUCSCORE plot'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-val'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='C'
displaygraph.label_y='ROC-AUC-SCORE'
displaygraph.Xdata = logregr_avgw2v.logrgr_lambda
displaygraph.ydatatrn = logregr_avgw2v.rocaucscoretrn
displaygraph.ydataval = logregr_avgw2v.rocaucscoreval
displaygraph.rocacuscoregraph()

#using the hyper parameter tuned value of lambda equal to 1
#perform logistic regression using test data
logregr_avgw2v.actualClasifier_logregrsn(1)


#displayig graphs for row-auc score for test data
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr BOW ROC Curve'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-test'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='False positive rate (1-Specificity)'
displaygraph.label_y='True positive rate (Sensitivity)'
displaygraph.constructgraph(logregr_avgw2v.roc_curve_test['fpr_trn'],logreg
r_avgw2v.roc_curve_test['tpr_trn'],\
                            logregr_avgw2v.roc_curve_test['fpr'],logregr_av
gw2v.roc_curve_test['tpr'])
# display the confusion matrix
data = [[logregr_avgw2v.confsnmtxytstpred['tn'] ,logregr_avgw2v.confsnmtxyt
stpred['fn']],[logregr_avgw2v.confsnmtxytstpred['fp'],logregr_avgw2v.confsn
mtxytstpred['tp']]]
displaygraph.draw_table(data)

#display the accuracy score
data1= [[logregr_avgw2v.accuracy_score_val,logregr_avgw2v.accuracy_score_te
st]]
displaygraph.draw_accscore(data1)
```

```
[0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.8460018878132247, 0.905236091839
7744, 0.9083179365968288, 0.9083325549541096, 0.9083273190194835, 0.9083264
757645549, 0.9083263633305643, 0.9083263885312863, 0.9083263788387008, 0.90
8326378838701, 0.908326378838701]
[0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.8244052533438749, 0.8683950
891272803, 0.8748570264256816, 0.8754673982050332, 0.8754626539838641, 0.87
54612060722087, 0.8754616373650421, 0.8754616373650423, 0.8754616065584113,
0.8754616065584113]
[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]
```



1



	Predicted: NO	Predicted: YES
Actual: NO	1036	529
Actual: YES	1925	16510

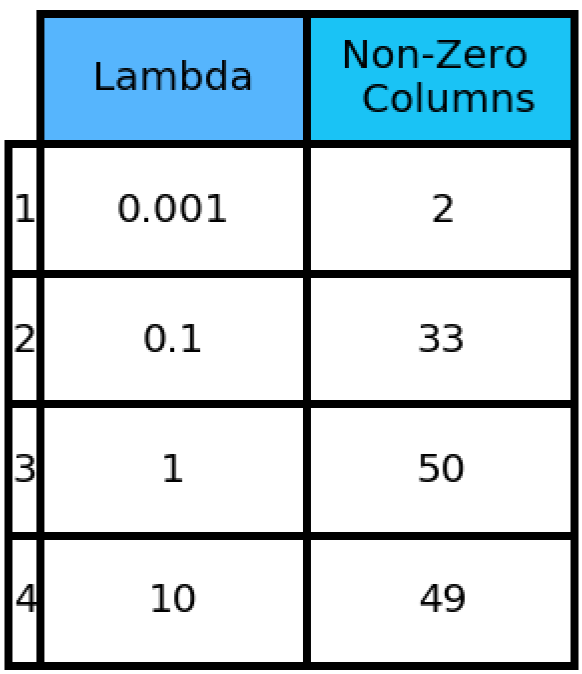
	Validation	Test
Accuracy Score	0.874125	0.8773

[5.3.1] Applying Logistic Regression with L1 regularization on AVG W2V SET 3

In [8]:
```
#set lambda and penalty
logregr_avgw2v.set_penaltyparm('l1')
logregr_avgw2v.set_lambdaparm(0.001)
#fit test data
logregr_avgw2v.logRegr_fitdata(logregr_avgw2v.xtest,logregr_avgw2v.ytest)
print(log_regr_avgw2v)
#get coefficients
w = log_regr_avgw2v.coef_
print(np.count_nonzero(w))

#set lambda and penalty
logregr_avgw2v.set_penaltyparm('l1')
logregr_avgw2v.set_lambdaparm(0.1)
#fit test data
logregr_avgw2v.logRegr_fitdata(logregr_avgw2v.xtest,logregr_avgw2v.ytest)
print(log_regr_avgw2v)
#get coefficients
w = log_regr_avgw2v.coef_
print(np.count_nonzero(w))

#set lambda and penalty
logregr_avgw2v.set_penaltyparm('l1')
logregr_avgw2v.set_lambdaparm(1)
#fit test data
logregr_avgw2v.logRegr_fitdata(logregr_avgw2v.xtest,logregr_avgw2v.ytest)
print(log_regr_avgw2v)
#get coefficients
w = log_regr_avgw2v.coef_
print(np.count_nonzero(w))

#set lambda and penalty
logregr_avgw2v.set_penaltyparm('l1')
logregr_avgw2v.set_lambdaparm(10)
#fit test data
logregr_avgw2v.logRegr_fitdata(logregr_avgw2v.xtest,logregr_avgw2v.ytest)
print(log_regr_avgw2v)
#get coefficients
w = log_regr_avgw2v.coef_
print(np.count_nonzero(w))
```

```
0.001
LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=Tr
ue,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
2
0.1
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
33
1
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
50
10
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
49
```

In [10]:
```
data2= [[0.001,2],[0.1,33],[1,50],[10,49]]
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.draw_sparsity(data2)
```



	Lambda	Non-Zero Columns
1	0.001	2
2	0.1	33
3	1	50
4	10	49

[5.3.2] Applying Logistic Regression with L2 regularization on AVG W2V, SET 3

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

In [76]:
```
logregr_avgw2v.set_penaltyparm('l2')
logregr_avgw2v.set_lambdaparm(1)
logregr_avgw2v.logRegr_fitdata(logregr_avgw2v.xtest,logregr_avgw2v.ytest)
print(log_regr_avgw2v)
#print(log_regr_tfidfwtw2v.coef_)
w = log_regr_avgw2v.coef_
print(np.count_nonzero(w))
```

```
1
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
50
```

[5.4] Logistic Regression on TFIDF W2V, SET 4

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

In [13]:
```
#instantiate logistic regression object and classifier
logregr_tfidfwtw2v = LogisticRegrsn()
log_regr_tfidfwtw2v = logregr_tfidfwtw2v.logRegrsn()

#load the data
logregr_tfidfwtw2v.xtrain,logregr_tfidfwtw2v.xtest,logregr_tfidfwtw2v.xval,
logregr_tfidfwtw2v.ytrain,logregr_tfidfwtw2v.ytest,logregr_tfidfwtw2v.yval
= logregr_tfidfwtw2v.load_data()

#instantiate the vectorizer
tfidf_model = TfidfVectorizer(ngram_range=(1,2), min_df=10, max_features=50
0)
tfidf_mtx_train = tfidf_model.fit_transform(logregr_tfidfwtw2v.xtrain)
tfidf_mtx_test = tfidf_model.fit_transform(logregr_tfidfwtw2v.xtest)
tfidf_mtx_val = tfidf_model.fit_transform(logregr_tfidfwtw2v.xval)

#print(list(tfidf_model.idf_))
print(type(tfidf_mtx_train))
print(tfidf_mtx_train.shape)
print(tfidf_mtx_test.shape)
print(tfidf_mtx_val.shape)
```

```
<class 'scipy.sparse.csr.csr_matrix'>
(64000, 500)
(20000, 500)
(16000, 500)
```

In [14]:

```
#create dictionary for the training/ test and validation set
dict_train = dict(zip(tfidf_model.get_feature_names(), list(tfidf_mtx_train
[0,:].nonzero()[1])))

dict_test = dict(zip(tfidf_model.get_feature_names(), list(tfidf_mtx_test[0
,:].nonzero()[1])))


dict_val = dict(zip(tfidf_model.get_feature_names(), list(tfidf_mtx_val[0
,:].nonzero()[1])))

#get feature names from the model
tfidf_feat = tfidf_model.get_feature_names()

#convert training data to list of sentences
lstsnt_xtrain=[]
lstsnt_xtest=[]
lstsnt_xval=[]

lstsnt_xtrain = logregr_tfidfwtw2v.listsent(logregr_tfidfwtw2v.xtrain)
lstsnt_xtest = logregr_tfidfwtw2v.listsent(logregr_tfidfwtw2v.xtest)
lstsnt_xval = logregr_tfidfwtw2v.listsent(logregr_tfidfwtw2v.xval)

#create the word to vec model
# min_count = 5 considers only words that occured atleast 5 times
w2v_mdl_xtrain=Word2Vec(lstsnt_xtrain,min_count=5,size=50, workers=4)
w2v_mdl_xtest=Word2Vec(lstsnt_xtest,min_count=5,size=50, workers=4)
w2v_mdl_xval=Word2Vec(lstsnt_xval,min_count=5,size=50, workers=4)

#get the vocabulary for the word to vec model
w2v_words_trn = list(w2v_mdl_xtrain.wv.vocab)
w2v_words_tst = list(w2v_mdl_xtest.wv.vocab)
w2v_words_val = list(w2v_mdl_xval.wv.vocab)
```

In [15]:

```
print(type(tfidf_feat),len(tfidf_feat))
print(type(lstsnt_xtrain),len(lstsnt_xtrain))
print(type(w2v_mdl_xtrain))
print(type(w2v_words_trn),len(w2v_words_trn))
print(type(dict_train))
```

```
<class 'list'> 500
<class 'list'> 64000
<class 'gensim.models.word2vec.Word2Vec'>
<class 'list'> 10881
<class 'dict'>
```

In [16]:
```
#function to create the tfidf weighted word to vec models
def tfidfwtw2v_crea(tfidf_feat, list_of_sentance, w2v_model,w2v_words,dicti
on):
    tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is st
ored 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/r
eview
        for word in sent: # for each word in a review/sentence
            if word in w2v_words and word in tfidf_feat and  word in dictio
n:
                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
                denom = sent.count(word)/len(sent)
                tf_idf = diction[word]*(denom)
                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
    return tfidf_sent_vectors
```

In [17]:
```
#creating the tfidf weighted word to vec models for training data
tfidfwtw2v_xtrain = tfidfwtw2v_crea(tfidf_feat,lstsnt_xtrain,w2v_mdl_xtrain
,w2v_words_trn,dict_train)
logregr_tfidfwtw2v.xtrain = tfidfwtw2v_xtrain
```

```
100%|██████████| 64000/64000 [01:15<00:00, 845.66it/s]
```

In [18]:
```
#creating the tfidf weighted word to vec models for test data
tfidfwtw2v_xtest = tfidfwtw2v_crea(tfidf_feat,lstsnt_xtest,w2v_mdl_xtest,w2
v_words_tst,dict_test)
logregr_tfidfwtw2v.xtest = tfidfwtw2v_xtest
```

```
100%|██████████| 20000/20000 [00:24<00:00, 816.55it/s]
```

In [19]:
```
#creating the tfidf weighted word to vec models for validation data
tfidfwtw2v_xval = tfidfwtw2v_crea(tfidf_feat,lstsnt_xval,w2v_mdl_xval,w2v_w
ords_val,dict_val)
logregr_tfidfwtw2v.xval = tfidfwtw2v_xval
```

```
100%|██████████| 16000/16000 [00:19<00:00, 818.41it/s]
```

In [36]:
```
return_hyparmtune = logregr_tfidfwtw2v.hyperparamtuning(wordvect.TFIDFAVG,[
0.00000000001,0.0000000001,0.0000000001,0.000000001,0.00000001,0.0000001,0.
000001,0.00001,0.0001,0.001,0.01,1,10,100,1000,10000,100000,1000000,1000000
0,100000000,1000000000,10000000000],'roc_auc',5,100,1)
```

```
Fitting 5 folds for each of 22 candidates, totalling 110 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.5698494192877963, total=   0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.5634991483456161, total=   0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.2s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................. C=1e-11, score=0.572171073696517, total=   0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.4s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.5617090595891954, total=   0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.6s remaining:
0.0s
[CV] C=1e-11 .........................................................
[CV] ................ C=1e-11, score=0.5600099296903074, total=   0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.7s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5698498553379417, total=   0.2s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    1.0s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5634991483456161, total=   0.1s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    1.2s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5721711706054327, total=   0.1s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    1.4s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5617090353491545, total=   0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.6s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5600099054502665, total=   0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    1.7s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5698498553379417, total=   0.1s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    1.9s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5634991483456161, total=   0.1s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    2.1s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5721711706054327, total=   0.1s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    2.4s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5617090353491545, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    2.5s remaining:
0.0s
[CV] C=1e-10 .........................................................
[CV] ................ C=1e-10, score=0.5600099054502665, total=   0.1s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:    2.8s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................. C=1e-09, score=0.569849637312869, total=   0.1s
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed:    2.9s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................ C=1e-09, score=0.5634991483456161, total=   0.1s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed:    3.2s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................ C=1e-09, score=0.5721710010148302, total=   0.1s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    3.4s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................. C=1e-09, score=0.561708986869073, total=   0.1s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed:    3.6s remaining:
0.0s
[CV] C=1e-09 .........................................................
[CV] ................ C=1e-09, score=0.5600096630498587, total=   0.1s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    3.9s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ................ C=1e-08, score=0.5698502187130629, total=   0.1s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    4.1s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ................ C=1e-08, score=0.5635048654475222, total=   0.1s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed:    4.3s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ................ C=1e-08, score=0.5721711706054327, total=   0.2s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:    4.7s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ................. C=1e-08, score=0.561709665590215, total=   0.2s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:    5.1s remaining:
0.0s
[CV] C=1e-08 .........................................................
[CV] ................ C=1e-08, score=0.5600096872898995, total=   0.1s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    5.4s remaining:
0.0s
[CV] C=1e-07 .........................................................
[CV] ................ C=1e-07, score=0.5697934837441474, total=   0.2s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    5.7s remaining:
0.0s
[CV] C=1e-07 .........................................................
[CV] ................ C=1e-07, score=0.5634955145944046, total=   0.3s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    6.1s remaining:
0.0s
[CV] C=1e-07 .........................................................
[CV] ................. C=1e-07, score=0.572173641782783, total=   0.2s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    6.5s remaining:
```

```
0.0s
[CV] C=1e-07 .........................................................
[CV] ................ C=1e-07, score=0.5616636579928044, total=   0.2s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    6.9s remaining:
0.0s
[CV] C=1e-07 .........................................................
[CV] ................. C=1e-07, score=0.560027018919061, total=   0.2s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    7.3s remaining:
0.0s
[CV] C=1e-06 .........................................................
[CV] ................ C=1e-06, score=0.5693619152252625, total=   0.3s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    7.7s remaining:
0.0s
[CV] C=1e-06 .........................................................
[CV] ................ C=1e-06, score=0.5629219148531663, total=   0.3s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    8.2s remaining:
0.0s
[CV] C=1e-06 .........................................................
[CV] ................ C=1e-06, score=0.5722543426823303, total=   0.2s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    8.4s remaining:
0.0s
[CV] C=1e-06 .........................................................
[CV] ................. C=1e-06, score=0.562223409014619, total=   0.3s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    8.9s remaining:
0.0s
[CV] C=1e-06 .........................................................
[CV] ................ C=1e-06, score=0.5596885067494939, total=   0.3s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    9.3s remaining:
0.0s
[CV] C=1e-05 .........................................................
[CV] ................ C=1e-05, score=0.5653016100715569, total=   0.3s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    9.8s remaining:
0.0s
[CV] C=1e-05 .........................................................
[CV] ................. C=1e-05, score=0.563587666525128, total=   0.4s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:   10.3s remaining:
0.0s
[CV] C=1e-05 .........................................................
[CV] .................. C=1e-05, score=0.57310445191806, total=   0.3s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:   10.9s remaining:
0.0s
[CV] C=1e-05 .........................................................
[CV] ................. C=1e-05, score=0.562271404295374, total=   0.4s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:   11.6s remaining:
0.0s
[CV] C=1e-05 .........................................................
[CV] ................. C=1e-05, score=0.563055036333882, total=   0.4s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:   12.2s remaining:
0.0s
[CV] C=0.0001 ........................................................
[CV] ............... C=0.0001, score=0.5633684059770265, total=   0.7s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:   13.1s remaining:
0.0s
[CV] C=0.0001 ........................................................
[CV] ............... C=0.0001, score=0.5784514531807146, total=   0.6s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:   13.8s remaining:
0.0s
```

```
[CV] C=0.0001 .........................................................
[CV] ............... C=0.0001, score=0.5834298303861394, total=   0.6s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:   14.5s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ................ C=0.0001, score=0.575530439964497, total=   0.6s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:   15.3s remaining:
0.0s
[CV] C=0.0001 .........................................................
[CV] ............... C=0.0001, score=0.5671863876851284, total=   0.6s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:   16.1s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.5635050350225786, total=   0.6s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:   16.9s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.5788939471782419, total=   0.8s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:   17.9s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................. C=0.001, score=0.582841205632191, total=   0.8s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:   18.8s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.5692951500332477, total=   0.5s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:   19.5s remaining:
0.0s
[CV] C=0.001 ..........................................................
[CV] ................ C=0.001, score=0.5678493285605664, total=   0.6s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:   20.3s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] ................. C=0.01, score=0.5647646869925467, total=   0.7s
[Parallel(n_jobs=1)]: Done  51 out of  51 | elapsed:   21.3s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] ................. C=0.01, score=0.5778285313230324, total=   0.8s
[Parallel(n_jobs=1)]: Done  52 out of  52 | elapsed:   22.2s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] ................. C=0.01, score=0.5800206231863496, total=   0.8s
[Parallel(n_jobs=1)]: Done  53 out of  53 | elapsed:   23.1s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] ................. C=0.01, score=0.5693550229339874, total=   0.7s
[Parallel(n_jobs=1)]: Done  54 out of  54 | elapsed:   24.0s remaining:
0.0s
[CV] C=0.01 ...........................................................
[CV] ................. C=0.01, score=0.5663523848418688, total=   0.7s
[Parallel(n_jobs=1)]: Done  55 out of  55 | elapsed:   24.8s remaining:
0.0s
[CV] C=1 ..............................................................
[CV] .................... C=1, score=0.5648212281613973, total=   0.7s
[Parallel(n_jobs=1)]: Done  56 out of  56 | elapsed:   25.6s remaining:
0.0s
[CV] C=1 ..............................................................
```

```
[CV] .................... C=1, score=0.5778167095190909, total=   0.8s
[Parallel(n_jobs=1)]: Done  57 out of  57 | elapsed:   26.6s remaining:
0.0s
[CV] C=1 .............................................................
[CV] .................... C=1, score=0.5800702890056448, total=   0.8s
[Parallel(n_jobs=1)]: Done  58 out of  58 | elapsed:   27.5s remaining:
0.0s
[CV] C=1 .............................................................
[CV] .................... C=1, score=0.5693587559002684, total=   0.7s
[Parallel(n_jobs=1)]: Done  59 out of  59 | elapsed:   28.3s remaining:
0.0s
[CV] C=1 .............................................................
[CV] .................... C=1, score=0.5663505910788507, total=   0.7s
[Parallel(n_jobs=1)]: Done  60 out of  60 | elapsed:   29.2s remaining:
0.0s
[CV] C=10 ............................................................
[CV] ................... C=10, score=0.5648212281613973, total=   0.6s
[Parallel(n_jobs=1)]: Done  61 out of  61 | elapsed:   29.9s remaining:
0.0s
[CV] C=10 ............................................................
[CV] .................... C=10, score=0.577816757969107, total=   0.9s
[Parallel(n_jobs=1)]: Done  62 out of  62 | elapsed:   31.0s remaining:
0.0s
[CV] C=10 ............................................................
[CV] ................... C=10, score=0.5800700951878133, total=   0.7s
[Parallel(n_jobs=1)]: Done  63 out of  63 | elapsed:   31.9s remaining:
0.0s
[CV] C=10 ............................................................
[CV] ................... C=10, score=0.5693588043803498, total=   0.8s
[Parallel(n_jobs=1)]: Done  64 out of  64 | elapsed:   32.9s remaining:
0.0s
[CV] C=10 ............................................................
[CV] ................... C=10, score=0.5663523606018278, total=   0.9s
[Parallel(n_jobs=1)]: Done  65 out of  65 | elapsed:   34.0s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] .................. C=100, score=0.5648211797113813, total=   0.9s
[Parallel(n_jobs=1)]: Done  66 out of  66 | elapsed:   35.1s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] ................... C=100, score=0.577816757969107, total=   0.7s
[Parallel(n_jobs=1)]: Done  67 out of  67 | elapsed:   36.0s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] ..................... C=100, score=0.5800701678695, total=   0.8s
[Parallel(n_jobs=1)]: Done  68 out of  68 | elapsed:   36.9s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] ................... C=100, score=0.569358780140309, total=   0.6s
[Parallel(n_jobs=1)]: Done  69 out of  69 | elapsed:   37.7s remaining:
0.0s
[CV] C=100 ...........................................................
[CV] .................. C=100, score=0.5663520939613793, total=   0.8s
[Parallel(n_jobs=1)]: Done  70 out of  70 | elapsed:   38.6s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.5648213250614298, total=   0.7s
```

```
[Parallel(n_jobs=1)]: Done  71 out of  71 | elapsed:   39.5s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.5778167095190909, total=   0.7s
[Parallel(n_jobs=1)]: Done  72 out of  72 | elapsed:   40.4s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.5800701436422712, total=   0.7s
[Parallel(n_jobs=1)]: Done  73 out of  73 | elapsed:   41.2s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.5693588043803499, total=   0.7s
[Parallel(n_jobs=1)]: Done  74 out of  74 | elapsed:   42.1s remaining:
0.0s
[CV] C=1000 ..........................................................
[CV] ................. C=1000, score=0.5663520939613793, total=   0.8s
[Parallel(n_jobs=1)]: Done  75 out of  75 | elapsed:   43.0s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.5648212523864056, total=   0.8s
[Parallel(n_jobs=1)]: Done  76 out of  76 | elapsed:   43.9s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................. C=10000, score=0.577816757969107, total=   0.7s
[Parallel(n_jobs=1)]: Done  77 out of  77 | elapsed:   44.8s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.5800701194150423, total=   0.8s
[Parallel(n_jobs=1)]: Done  78 out of  78 | elapsed:   45.7s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.5693587801403092, total=   0.6s
[Parallel(n_jobs=1)]: Done  79 out of  79 | elapsed:   46.5s remaining:
0.0s
[CV] C=10000 .........................................................
[CV] ................ C=10000, score=0.5663520939613793, total=   0.8s
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:   47.4s remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.5648212766114135, total=   0.7s
[Parallel(n_jobs=1)]: Done  81 out of  81 | elapsed:   48.3s remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ................ C=100000, score=0.577816757969107, total=   0.7s
[Parallel(n_jobs=1)]: Done  82 out of  82 | elapsed:   49.2s remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.5800702647784158, total=   0.7s
[Parallel(n_jobs=1)]: Done  83 out of  83 | elapsed:   50.1s remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ................ C=100000, score=0.569358780140309, total=   0.6s
[Parallel(n_jobs=1)]: Done  84 out of  84 | elapsed:   50.9s remaining:
0.0s
[CV] C=100000 ........................................................
[CV] ............... C=100000, score=0.5663520939613793, total=   0.8s
[Parallel(n_jobs=1)]: Done  85 out of  85 | elapsed:   51.8s remaining:
```

```
0.0s
[CV] C=1000000 .......................................................
[CV] .............. C=1000000, score=0.5648213250614298, total=   0.7s
[Parallel(n_jobs=1)]: Done  86 out of  86 | elapsed:   52.7s remaining:
0.0s
[CV] C=1000000 .......................................................
[CV] ............... C=1000000, score=0.577816757969107, total=   0.7s
[Parallel(n_jobs=1)]: Done  87 out of  87 | elapsed:   53.6s remaining:
0.0s
[CV] C=1000000 .......................................................
[CV] ................. C=1000000, score=0.5800701678695, total=   0.8s
[Parallel(n_jobs=1)]: Done  88 out of  88 | elapsed:   54.5s remaining:
0.0s
[CV] C=1000000 .......................................................
[CV] .............. C=1000000, score=0.5693588043803499, total=   0.6s
[Parallel(n_jobs=1)]: Done  89 out of  89 | elapsed:   55.3s remaining:
0.0s
[CV] C=1000000 .......................................................
[CV] .............. C=1000000, score=0.5663520939613793, total=   0.7s
[Parallel(n_jobs=1)]: Done  90 out of  90 | elapsed:   56.2s remaining:
0.0s
[CV] C=10000000 ......................................................
[CV] ............. C=10000000, score=0.5648213250614298, total=   0.7s
[Parallel(n_jobs=1)]: Done  91 out of  91 | elapsed:   57.1s remaining:
0.0s
[CV] C=10000000 ......................................................
[CV] .............. C=10000000, score=0.577816757969107, total=   0.7s
[Parallel(n_jobs=1)]: Done  92 out of  92 | elapsed:   58.0s remaining:
0.0s
[CV] C=10000000 ......................................................
[CV] ................ C=10000000, score=0.5800701678695, total=   0.7s
[Parallel(n_jobs=1)]: Done  93 out of  93 | elapsed:   58.9s remaining:
0.0s
[CV] C=10000000 ......................................................
[CV] ............. C=10000000, score=0.5693587559002684, total=   0.5s
[Parallel(n_jobs=1)]: Done  94 out of  94 | elapsed:   59.5s remaining:
0.0s
[CV] C=10000000 ......................................................
[CV] ............. C=10000000, score=0.5663523606018279, total=   0.7s
[Parallel(n_jobs=1)]: Done  95 out of  95 | elapsed:  1.0min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ............ C=100000000, score=0.5648213250614298, total=   0.7s
[Parallel(n_jobs=1)]: Done  96 out of  96 | elapsed:  1.0min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ............. C=100000000, score=0.577816757969107, total=   0.6s
[Parallel(n_jobs=1)]: Done  97 out of  97 | elapsed:  1.0min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ............ C=100000000, score=0.5800701436422712, total=   0.7s
[Parallel(n_jobs=1)]: Done  98 out of  98 | elapsed:  1.0min remaining:
0.0s
[CV] C=100000000 .....................................................
[CV] ............ C=100000000, score=0.5693587316602275, total=   0.6s
[Parallel(n_jobs=1)]: Done  99 out of  99 | elapsed:  1.1min remaining:
0.0s
```

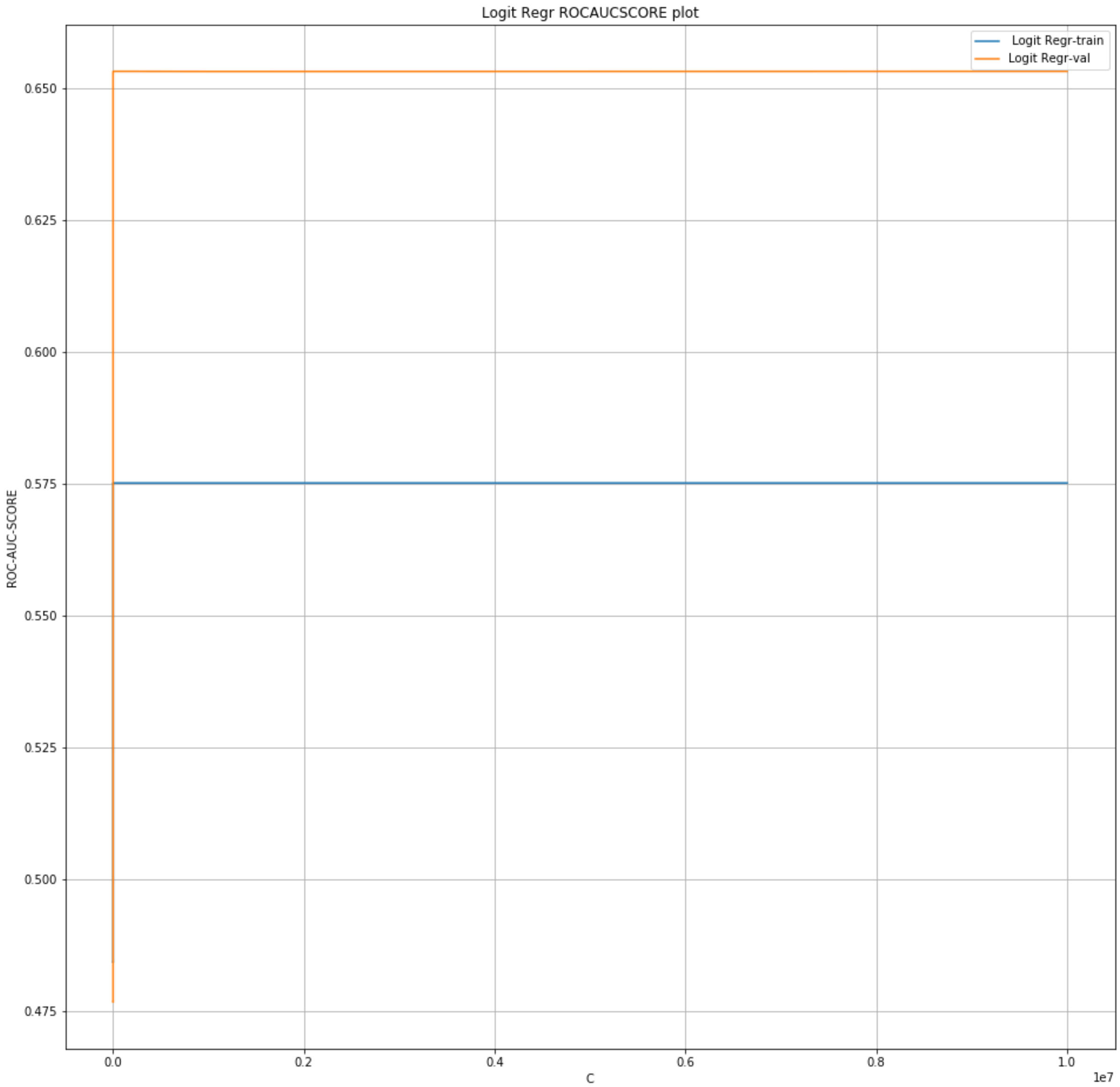
```
[CV] C=100000000 .....................................................
[CV] ........... C=100000000, score=0.5663520939613793, total=   0.8s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.5648213250614298, total=   0.8s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.5778167095190909, total=   0.7s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.5800701436422712, total=   0.7s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.5693587316602275, total=   0.6s
[CV] C=1000000000 ....................................................
[CV] .......... C=1000000000, score=0.5663520939613793, total=   0.7s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.5648213008364216, total=   0.7s
[CV] C=10000000000 ...................................................
[CV] .......... C=10000000000, score=0.577816757969107, total=   0.7s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.5800701436422712, total=   0.8s
[CV] C=10000000000 ...................................................
[CV] .......... C=10000000000, score=0.569358780140309, total=   0.7s
[CV] C=10000000000 ...................................................
[CV] ......... C=10000000000, score=0.5663520939613793, total=   0.8s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:  1.2min finished
```

In [20]:
```
print(return_hyparmtune[0],return_hyparmtune[1],return_hyparmtune[2])

#hyper parameter results 0.5735932894235685 {'C': 0.0001}

#generate multiple rocauc scores by varying lambda
logregr_tfidfwtw2v.calcrocaucscore_logregrsn(10000000)

print(len(logregr_tfidfwtw2v.rocaucscoretrn))
print(len(logregr_tfidfwtw2v.rocaucscoreval))
print(len(logregr_tfidfwtw2v.logrgr_lambda))

#plot graph of the rocauc scores
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr ROCAUCSCORE plot'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-val'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='C'
displaygraph.label_y='ROC-AUC-SCORE'
displaygraph.Xdata = logregr_tfidfwtw2v.logrgr_lambda
displaygraph.ydatatrn = logregr_tfidfwtw2v.rocaucscoretrn[:19]
displaygraph.ydataval = logregr_tfidfwtw2v.rocaucscoreval
displaygraph.rocacuscoregraph()

#using hyperparameter value process logistic regression using test data
logregr_tfidfwtw2v.actualClasifier_logregrsn(0.0001)


#plot the rocauc scores
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.graph_title='Logit Regr BOW ROC Curve'
displaygraph.legnd_1 = ' Logit Regr-train'
displaygraph.legnd_2 = 'Logit Regr-test'
displaygraph.graph_parameters['show_legnd']= True
displaygraph.label_x='False positive rate (1-Specificity)'
displaygraph.label_y='True positive rate (Sensitivity)'
displaygraph.constructgraph(logregr_tfidfwtw2v.roc_curve_test['fpr_trn'],lo
gregr_tfidfwtw2v.roc_curve_test['tpr_trn'],\
                            logregr_tfidfwtw2v.roc_curve_test['fpr'],logreg
r_tfidfwtw2v.roc_curve_test['tpr'])

#display the confusion matrix
data = [[logregr_tfidfwtw2v.confsnmtxytstpred['tn'] ,logregr_tfidfwtw2v.con
fsnmtxytstpred['fn']],[logregr_tfidfwtw2v.confsnmtxytstpred['fp'],logregr_t
fidfwtw2v.confsnmtxytstpred['tp']]]
displaygraph.draw_table(data)

#display accuracy score
data1= [[logregr_tfidfwtw2v.accuracy_score_val,logregr_tfidfwtw2v.accuracy_
score_test]]
displaygraph.draw_accscore(data1)
```

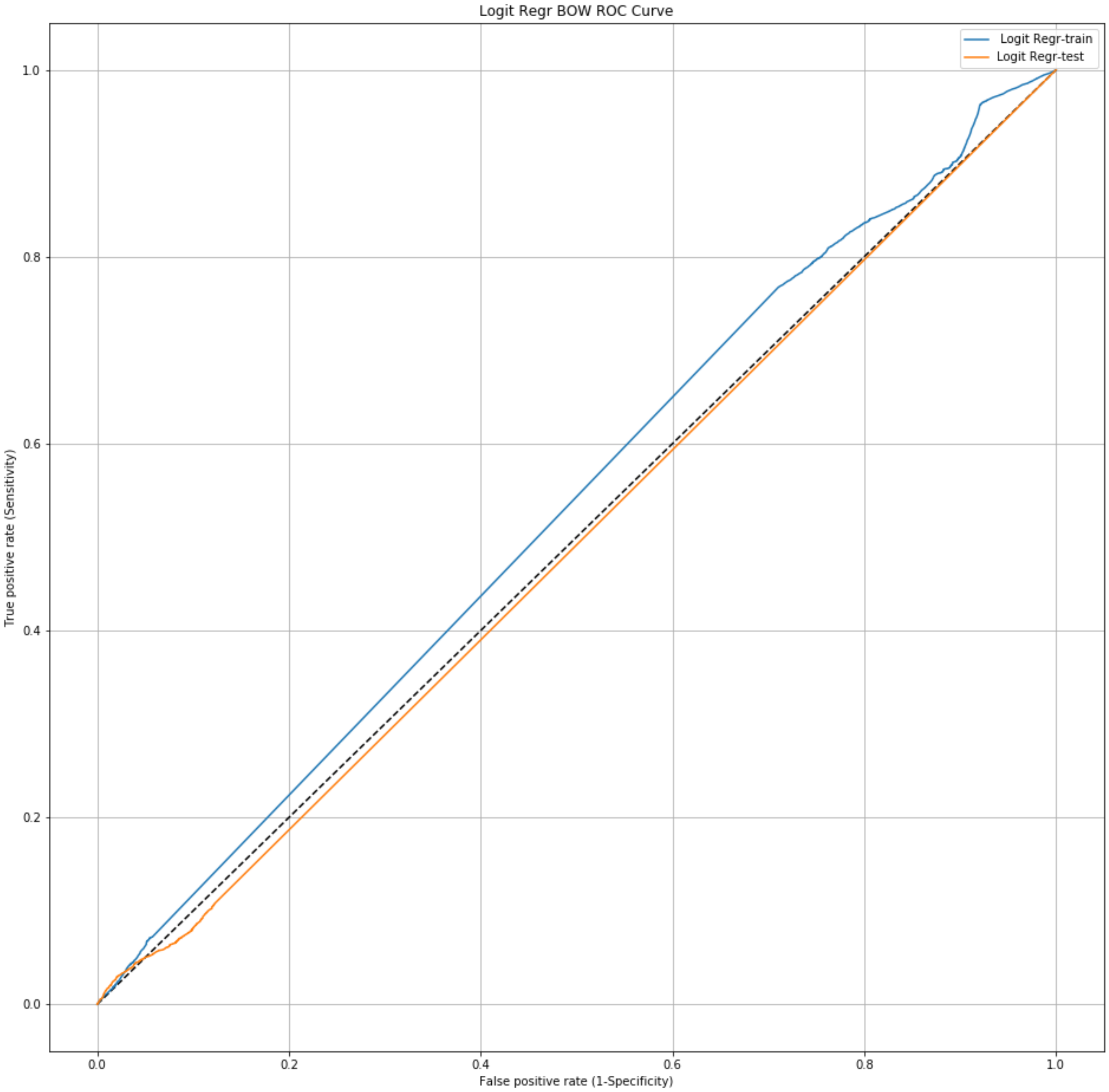
```
1e-11
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
9.999999999999999e-11
Fitting probability generation and roc 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...
9.999999999999999e-10
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...
9.999999999999999e-09
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...
9.999999999999998e-08
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...
9.999999999999997e-07
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...
9.999999999999997e-06
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...
9.999999999999998e-05
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...
0.0009999999999999998
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...
0.009999999999999998
Fitting probability generation and roc auc score generation for training da
ta complete...
```

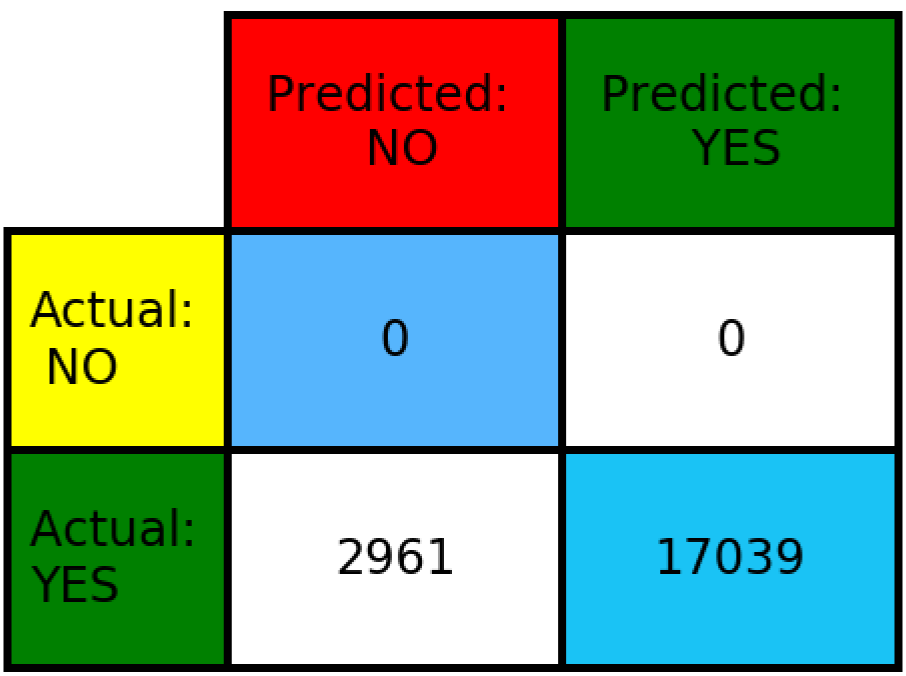
```
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
0.09999999999999998
Fitting probability generation and roc auc score generation for training da
ta complete...
Fitting probability generation and roc auc score generation for validation
data complete...
Predicting labels for training data complete...
0.9999999999999998
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...
9.999999999999998
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...
99.99999999999999
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...
999.9999999999999
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...
9999.999999999998
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...
99999.99999999999
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...
999999.9999999999
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...
9999999.999999998
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...
19
19
19
```




```
0.0001
```





	Predicted: NO	Predicted: YES
Actual: NO	0	0
Actual: YES	2961	17039

	Validation	Test
Accuracy Score	0.8501875	0.85195

[5.4.1] Applying Logistic Regression with L1 regularization on TFIDF W2V, SET 4

In [21]:
```
#set lambda and penalty
logregr_tfidfwtw2v.set_penaltyparm('l1')
logregr_tfidfwtw2v.set_lambdaparm(0.001)
#fit test data
logregr_tfidfwtw2v.logRegr_fitdata(logregr_tfidfwtw2v.xtest,logregr_tfidfwt
w2v.ytest)
print(log_regr_tfidfwtw2v)
#get coefficients
w = log_regr_tfidfwtw2v.coef_
print(np.count_nonzero(w))

#set lambda and penalty
logregr_tfidfwtw2v.set_penaltyparm('l1')
logregr_tfidfwtw2v.set_lambdaparm(0.1)
#fit test data
logregr_tfidfwtw2v.logRegr_fitdata(logregr_tfidfwtw2v.xtest,logregr_tfidfwt
w2v.ytest)
print(log_regr_tfidfwtw2v)
#get coefficients
w = log_regr_tfidfwtw2v.coef_
print(np.count_nonzero(w))

#set lambda and penalty
logregr_tfidfwtw2v.set_penaltyparm('l1')
logregr_tfidfwtw2v.set_lambdaparm(1)
#fit test data
logregr_tfidfwtw2v.logRegr_fitdata(logregr_tfidfwtw2v.xtest,logregr_tfidfwt
w2v.ytest)
print(log_regr_tfidfwtw2v)
#get coefficients
w = log_regr_tfidfwtw2v.coef_
print(np.count_nonzero(w))

#set lambda and penalty
logregr_tfidfwtw2v.set_penaltyparm('l1')
logregr_tfidfwtw2v.set_lambdaparm(10)
#fit test data
logregr_tfidfwtw2v.logRegr_fitdata(logregr_tfidfwtw2v.xtest,logregr_tfidfwt
w2v.ytest)
print(log_regr_tfidfwtw2v)
#get coefficients
w = log_regr_tfidfwtw2v.coef_
print(np.count_nonzero(w))
```

```
0.001
LogisticRegression(C=0.001, class_weight=None, dual=False, fit_intercept=Tr
ue,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
0
0.1
LogisticRegression(C=0.1, class_weight=None, dual=False, fit_intercept=Tru
e,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
4
1
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
8
10
LogisticRegression(C=10, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=200, multi_class='warn',
          n_jobs=None, penalty='l1', random_state=42, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
8
```

In [22]:

```
data2= [[0.001,0],[0.1,4],[1,8],[10,8]]
displaygraph = drawgraphs()
displaygraph.setdefaultparm()
displaygraph.draw_sparsity(data2)
```

	Lambda	Non-Zero Columns
1	0.001	0
2	0.1	4
3	1	8
4	10	8

[5.4.2] Applying Logistic Regression with L2 regularization on TFIDF W2V, SET 4

In [50]:
```
logregr_tfidfwtw2v.set_penaltyparm('l2')
logregr_tfidfwtw2v.set_lambdaparm(0.0001)
logregr_tfidfwtw2v.logRegr_fitdata(logregr_tfidfwtw2v.xtest,logregr_tfidfwt
w2v.ytest)
print(log_regr_tfidfwtw2v)
#print(log_regr_tfidfwtw2v.coef_)
w1 = log_regr_tfidfwtw2v.coef_
print(np.count_nonzero(w1))
```

```
0.0001
LogisticRegression(C=0.0001, class_weight=None, dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=200,
          multi_class='warn', n_jobs=None, penalty='l2', random_state=None,
          solver='warn', tol=0.0001, verbose=0, warm_start=False)
50
```

[6] Conclusions

In [0]:
```
# Please compare all your models using Prettytable library
```

In [27]:

```
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>'
L14  =  '<th>Model </th>'
L15  =  '<th>Hyper Parameter</th>'
L16  =  '<th>AUC</th>'
L17  =  '</tr>'
L18  =  '<tr bgcolor="black"><td> BOW </td><td> Logistic Regression </td><t
d> 1 </td><td> 0.9233</td></tr>'
L19  =  '<tr><td> TFIDF </td><td> Logistic Regression </td><td> 10 </td><td
> 0.9938</td></tr>'
L20  =  '<tr><td> W2V </td><td> Logistic Regression </td><td> 1 </td><td>
 0.8773</td></tr>'
L21  =  '<tr><td> TFIDFW2V </td><td> Logistic Regression </td><td> 0.0001
 </td><td> 0.8520</td></tr>'

L22  =  '</table>'

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

html_string = L1+L2+L3+L4+L5+L6+L7+L8+L9+L10+L11+L12+L13+L14+L15+L16+L17+L1
8+L19+L20+L21+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 |        Model        | Hyper Parameter |  AUC   |
+------------+---------------------+-----------------+--------+
|    BOW     | Logistic Regression |        1        | 0.9233 |
|   TFIDF    | Logistic Regression |        10       | 0.9938 |
|    W2V     | Logistic Regression |        1        | 0.8773 |
|  TFIDFW2V  | Logistic Regression |      0.0001     | 0.8520 |
+------------+---------------------+-----------------+--------+
```

In [26]:

```
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: #000000; color: white;}'
L7  =  '--->'
L8  =  '</STYLE>'

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

L11  =  '<table border=1 solid> '
L12  =  '<tr>'
L13  =  '<th>Lambda </th>'
L14  =  '<th>Bag Of words </th>'
L15  =  '<th>Tf-IDF</th>'
L16  =  '<th>Avgw2v</th>'
L17  =  '<th>TfIdf-wt-w2v</th></tr>'
L18  =  '<tr bgcolor="black"><td> 0.001 </td><td> 0 </td><td> 0 </td><td> 2 </td><td> 0</td></tr>'
L19  =  '<tr><td> 0.1 </td><td> 369 </td><td> 28 </td><td> 33</td><td> 4</td></tr>'
L20  =  '<tr><td> 1 </td><td> 882 </td><td> 474 </td><td> 50</td><td> 8</td></tr>'
L21  =  '<tr><td> 10 </td><td> 987 </td><td> 3182 </td><td> 49</td><td> 8</td></tr>'

L22  =  '</table>'

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

html_string = L1+L2+L3+L4+L5+L6+L7+L8+L9+L10+L11+L12+L13+L14+L15+L16+L17+L18+L19+L20+L21+L22+L23+L24

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

print(tbl)
```

```
+--------+--------------+--------+--------+--------------+
| Lambda | Bag Of words | Tf-IDF | Avgw2v | TfIdf-wt-w2v |
+--------+--------------+--------+--------+--------------+
| 0.001  |      0       |   0    |   2    |      0       |
|  0.1   |     369      |   28   |   33   |      4       |
|   1    |     882      |  474   |   50   |      8       |
|   10   |     987      |  3182  |   49   |      8       |
+--------+--------------+--------+--------+--------------+
```