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 [3]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
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 [4]:
```
# using SQLite Table to read data.
con = sqlite3.connect('database.sqlite')

# 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 power

# 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 25000""", 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 (25000, 10)
```

Out[4]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	1	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	0	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	1	1219017600	"Delight" says it all	This is a confection that has been around a fe...

In [5]:
```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [6]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[6]:

	UserId	ProductId	ProfileName	Time	Score	Text	COUNT(*)
0	#oc-R115TNMSPFT9I7	B007Y59HVM	Breyton	1331510400	2	Overall its just OK when considering the price...	2
1	#oc-R11D9D7SHXIJB9	B005HG9ET0	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B007Y59HVM	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ET0	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBE1U	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

In [7]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[7]:

	UserId	ProductId	ProfileName	Time	Score	Text	COUNT(*)
80638	AZY10LLTJ71NX	B006P7E5ZI	undertheshrine "undertheshrine"	1334707200	5	I was recommended to try green tea extract to ...	5

In [8]:
```
display['COUNT(*)'].sum()
```

Out[8]:
```
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 [9]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[9]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	78445	B000HDL1RQ	AR5J8UI46CURR	Geetha Krishnan	2	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
1	138317	B000HDOPYC	AR5J8UI46CURR	Geetha Krishnan	2	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
2	138277	B000HDOPYM	AR5J8UI46CURR	Geetha Krishnan	2	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
3	73791	B000HDOPZG	AR5J8UI46CURR	Geetha Krishnan	2	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
4	155049	B000PAQ75C	AR5J8UI46CURR	Geetha Krishnan	2	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...

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 [10]:
```
#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 [11]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first', inplace=False)
final.shape
```

Out[11]:
```
(23953, 10)
```

In [12]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[12]:
```
95.812
```

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 [13]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[13]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	1	5	1224892800	Bought This for My Son at College	My son loves spaghetti so I didn't hesitate or...
1	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	2	4	1212883200	Pure cocoa taste with crunchy almonds inside	It was almost a 'love at first bite' - the per...

In [14]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [15]:
```
#Before starting the next phase of preprocessing lets see the number of entries left
print(final.shape)

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

```
(23953, 10)
```

Out[15]:
```
1    20071
0     3882
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 [16]:
```
# printing some random reviews
sent_0 = final['Text'].values[0]
print(sent_0)
print("="*50)

sent_1000 = final['Text'].values[100]
print(sent_1000)
print("="*50)

sent_1500 = final['Text'].values[150]
print(sent_1500)
print("="*50)

sent_4900 = final['Text'].values[490]
print(sent_4900)
print("="*50)
```

```
My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its very hard to find any chicken products made in the USA but they are out there, but this one isnt.  Its too bad too because its a good product but I wont take any chances till they know what is going on with the china imports.
==================================================
After reading the book "Deadly Feasts" I was horrified to learn what kinds of items may end up in my dogs' food! I started researching different dog foods and consistently received the same brand recommendations: Innova, Canidae, Wellness, Chicken Soup for the Dog Lover's Soul...etc. I initially settled on Canidae, but also tried Wellness on a recommendation. The Wellness produced loose stools in two of my foster poodles, so I no longer use that brand. The Canidae, however, is perfect for my multi dog household, which often includes dogs ranging from puppyhood to senior status. The ingredients are phenomenal and, as all reviewers have said, the food contains no corn, wheat or soy, all common allergens in dogs. Although I also use and would recommend Innova Evo dry dog food, Canidae costs $35 for 40 pounds and Innova costs about $50 for a 28 pound bag. So, if you are budgeting a bit but would still like to give your dog the best, Canidae is great. After switching my dogs to Canidae, I noticed several things. First, my labrador retriever's coat sheen increased to the point of nearly blinding me in the sunglight; second, he lost about 10 pounds and developed greater muscle tone (although he was always fed recommended portions of commercial foods); third, my shedding dogs shed a lot less than in the past; and fourth, bad doggy breath is greatly reduced. I see no excuse not to switch your dog from a commercial food to Canidae or another super premium brand - it costs no more than a fancy bag of Iams or Science Diet, but can add quality years to your pets' lives! If you own a dog prone to allergies, such as a poodle, this food can't be beat. I had a family adopt a 5-pound malipoo from me and they emailed me to say they cannot feed him anything other than Canidae or he develops loose stools and needs to be taken out to the bathroom contantly. For your dog's sake switch their food; you can do a lot better than chain-store foods!
==================================================
Other than a few table scraps, Canidae is the only brand (dry and wet) we feed to our blk lab.  She's done excellent on it and has a very healthy and shinny coat.  People routinely comment how nice her coat looks, most even ask what we feed.  I sent a bag to my brother and he too noticed his dog's coat was healthier looking after a week or two.  I highly recommend this product.
==================================================
This is a very good snack and good for you. Price was good. Shipping not bad, but good for distance.
==================================================
```

In [17]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/4084039
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)
```

```
My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its very hard to find any chicken products made in the USA but they are out there, but this one isnt.  Its too bad too because its a good product but I wont take any chances till they know what is going on with the china imports.
```

In [18]:
```
# 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)
```

```
My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its very hard to find any chicken products made in the USA but they are out there, but this one isnt.  Its too bad too because its a good product but I wont take any chances till they know what is going on with the china imports.
==================================================
After reading the book "Deadly Feasts" I was horrified to learn what kinds of items may end up in my dogs' food! I started researching different dog foods and consistently received the same brand recommendations: Innova, Canidae, Wellness, Chicken Soup for the Dog Lover's Soul...etc. I initially settled on Canidae, but also tried Wellness on a recommendation. The Wellness produced loose stools in two of my foster poodles, so I no longer use that brand. The Canidae, however, is perfect for my multi dog household, which often includes dogs ranging from puppyhood to senior status. The ingredients are phenomenal and, as all reviewers have said, the food contains no corn, wheat or soy, all common allergens in dogs. Although I also use and would recommend Innova Evo dry dog food, Canidae costs $35 for 40 pounds and Innova costs about $50 for a 28 pound bag. So, if you are budgeting a bit but would still like to give your dog the best, Canidae is great. After switching my dogs to Canidae, I noticed several things. First, my labrador retriever's coat sheen increased to the point of nearly blinding me in the sunglight; second, he lost about 10 pounds and developed greater muscle tone (although he was always fed recommended portions of commercial foods); third, my shedding dogs shed a lot less than in the past; and fourth, bad doggy breath is greatly reduced. I see no excuse not to switch your dog from a commercial food to Canidae or another super premium brand - it costs no more than a fancy bag of Iams or Science Diet, but can add quality years to your pets' lives! If you own a dog prone to allergies, such as a poodle, this food can't be beat. I had a family adopt a 5-pound malipoo from me and they emailed me to say they cannot feed him anything other than Canidae or he develops loose stools and needs to be taken out to the bathroom contantly. For your dog's sake switch their food; you can do a lot better than chain-store foods!
==================================================
Other than a few table scraps, Canidae is the only brand (dry and wet) we feed to our blk lab.  She's done excellent on it and has a very healthy and shinny coat.  People routinely comment how nice her coat looks, most even ask what we feed.  I sent a bag to my brother and he too noticed his dog's coat was healthier looking after a week or two.  I highly recommend this product.
==================================================
This is a very good snack and good for you. Price was good. Shipping not bad, but good for distance.
```

In [19]:
```
# 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 [20]:
```
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
Other than a few table scraps, Canidae is the only brand (dry and wet) we feed to our blk lab.  She is done excellent on it and has a very healthy and shinny coat.  People routinely comment how nice her coat looks, most even ask what we feed.  I sent a bag to my brother and he too noticed his dog is coat was healthier looking after a week or two.  I highly recommend this product.
==================================================
```

In [21]:
```
#remove words with numbers python: https://stackoverflow.com/a/18082370/4084039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
My dogs loves this chicken but its a product from China, so we wont be buying it anymore.  Its very hard to find any chicken products made in the USA but they are out there, but this one isnt.  Its too bad too because its a good product but I wont take any chances till they know what is going on with the china imports.
```

In [22]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
print(sent_1500)
```

```
Other than a few table scraps Canidae is the only brand dry and wet we feed to our blk lab She is done excellent on it and has a very healthy and shinny coat People routinely comment how nice her coat looks most even ask what we feed I sent a bag to my brother and he too noticed his dog is coat was healthier looking after a week or two I highly recommend this product
```

In [23]:
```
# 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', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', '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', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "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', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

In [24]:
```
# 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() not in stopwords)
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 23953/23953 [00:24<00:00, 993.24it/s]
```

In [25]:
```
preprocessed_reviews[150]
```

Out[25]:
```
'table scraps canidae brand dry wet feed blk lab done excellent healthy shinny coat people routinely comment nice coat looks even ask feed sent bag brother noticed dog coat healthier looking week two highly recommend product'
```

[3.2] Preprocessing Review Summary

In [26]:
```
## Similartly you can do preprocessing for review summary also.
```

[3.2] Preprocessing Review Summary

In [27]:
```
from sklearn.cross_validation import train_test_split
X = preprocessed_reviews
Y = final['Score']
X_1 , X_test , Y_1 , Y_test  = train_test_split(X,Y,test_size=0.2,random_state=0)
X_tr , X_cv , Y_tr , Y_cv  = train_test_split(X_1,Y_1,test_size=0.4,random_state=0)
```

```
C:\Users\RajMahendra\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
```

[4] Featurization

[4.1] BAG OF WORDS

In [28]:
```
#BoW
count_vect = CountVectorizer() #in scikit-learn
count_vect.fit(preprocessed_reviews)
print("some feature names ", count_vect.get_feature_names()[:10])
print('='*50)

X_Bow_Tr = count_vect.transform(X_tr)
X_Bow_Cv = count_vect.transform(X_cv)
X_Bow_Test = count_vect.transform(X_test)


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', 'aaa', 'aaaa', 'aaaaa', 'aaaaaaaaaaa', 'aaaand', 'aaah', 'aadp', 'aafco', 'aahhhs']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (23953, 28671)
the number of unique words  28671
```

[4.2] Bi-Grams and n-Grams.

bi-gram, tri-gram and n-gram #removing stop words like "not" should be avoided before building n-grams # count_vect = CountVectorizer(ngram_range=(1,2)) # please do read the CountVectorizer documentation http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html # you can choose these numebrs min_df=10, max_features=5000, of your choice count_vect = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000) final_bigram_counts = count_vect.fit_transform(preprocessed_reviews) print("the type of count vectorizer ",type(final_bigram_counts)) print("the shape of out text BOW vectorizer ",final_bigram_counts.get_shape()) print("the number of unique words including both unigrams and bigrams ", final_bigram_counts.get_shape()[1])

In [29]:
```
Bow_Feature = count_vect.get_feature_names()
```

In [30]:
```
X_Bow_Tr   = X_Bow_Tr.toarray()
X_Bow_Cv   = X_Bow_Cv.toarray()
X_Bow_Test = X_Bow_Test.toarray()
```

[4.3] TF-IDF

In [31]:
```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=20, max_df=50)
tf_idf_vect.fit(X_tr)
print("some sample features(unique words in the corpus)",tf_idf_vect.get_feature_names()[0:10])
print('='*50)


X_Tfidf_Tr = tf_idf_vect.transform(X_tr)
X_Tfidf_Cv = tf_idf_vect.transform(X_cv)
X_Tfidf_Test = tf_idf_vect.transform(X_test)

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

```
some sample features(unique words in the corpus) ['ability', 'able find', 'able get', 'absolute favorite', 'absolutely delicious', 'absolutely loves', 'absolutely no', 'acceptable', 'according', 'acid coffee']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text TFIDF vectorizer  (11497, 2072)
the number of unique words including both unigrams and bigrams  2072
```

In [32]:
```
X_Tfidf_Tr = X_Tfidf_Tr.toarray()
```

In [33]:
```
X_Tfidf_Cv = X_Tfidf_Cv.toarray()
```

In [34]:
```
X_Tfidf_Test = X_Tfidf_Test.toarray()
```

In [35]:
```
tf_idf_feature = tf_idf_vect.get_feature_names()
```

[4.4] Word2Vec

In [36]:
```
# 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 [37]:
```
# 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/#.W17SRFAzZPY
# 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-negative300.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_w2v = True, to train your own w2v ")
```

```
[('awesome', 0.8308538198471069), ('good', 0.8261924386024475), ('excellent', 0.7953504323959351), ('fantastic', 0.7770377397537231), ('wonderful', 0.7632495760917664), ('amazing', 0.7553866505622864), ('perfect', 0.7228683829307556), ('decent', 0.6787499785423279), ('delicious', 0.6703792810440063), ('terrific', 0.669090986251831)]
==================================================
[('best', 0.7614737749099731), ('closest', 0.7282196283340454), ('awful', 0.7086548805236816), ('ever', 0.7062759399414062), ('tastiest', 0.7053784728050232), ('world', 0.704605221748352), ('addicted', 0.7003800868988037), ('consensus', 0.6876540780067444), ('greatest', 0.6873849630355835), ('foul', 0.6813795566558838)]
```

In [38]:
```
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  9295
sample words  ['dogs', 'loves', 'chicken', 'product', 'china', 'wont', 'buying', 'anymore', 'hard', 'find', 'products', 'made', 'usa', 'one', 'isnt', 'bad', 'good', 'take', 'chances', 'till', 'know', 'going', 'imports', 'love', 'saw', 'pet', 'store', 'tag', 'attached', 'regarding', 'satisfied', 'safe', 'used', 'fly', 'bait', 'seasons', 'ca', 'not', 'beat', 'great', 'available', 'traps', 'unreal', 'course', 'total', 'pretty', 'stinky', 'right', 'nearby', 'received']
```

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

[4.4.1.1] Avg W2v

In [39]:
```
# average Word2Vec
# compute average word2vec for each review.

def getAvgWordToVector(list_of_sentance):
    sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
    #print("list_of_sentance>>",list_of_sentance)
    for sentence in list_of_sentance: # for each review/sentence
        #print("sentence>>",sentence)
        sent = sentence.split()
        #print("sent>>",sent)
        sent_vec = np.zeros(50) # as word vectors are of zero length 50, 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 sentence/review
        for word in sent: # for each word in a review/sentence
            if word in w2v_words:
                #print("word>>",word)
                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
```

In [40]:
```
X_AvgW2V_Tr     = getAvgWordToVector(X_tr)
```

In [41]:
```
X_AvgW2V_Cv     = getAvgWordToVector(X_cv)
```

In [42]:
```
X_AvgW2V_Test   = getAvgWordToVector(X_test)
```

[4.4.1.2] TFIDF weighted W2v

In [43]:
```
# 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 [44]:
```
# 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_val = tfidf

def getAvgW2VtfIdfToVector(list_of_sentance):
    tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored in this list
    row=0;
    for sentence in list_of_sentance: # for each review/sentence
        sent = []
        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/review
        sent = sentence.split()
        for word in sent: # for each word in a review/sentence3
            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
    return tfidf_sent_vectors
```

In [45]:
```
X_AvgW2VtfIdf_Tr     = getAvgW2VtfIdfToVector(X_tr)
X_AvgW2VtfIdf_Cv     = getAvgW2VtfIdfToVector(X_cv)
X_AvgW2VtfIdf_Test   = getAvgW2VtfIdfToVector(X_test)
```

[5] Assignment 7: SVM

1. **Apply SVM 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. **Procedure**
   - You need to work with 2 versions of SVM
     - Linear kernel
     - RBF kernel
   - When you are working with linear kernel, use SGDClassifier' with hinge loss because it is computationally less expensive.
   - When you are working with 'SGDClassifier' with hinge loss and trying to find the AUC score, you would have to use CalibratedClassifierCV (https://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html)
   - Similarly, like kdtree of knn, when you are working with RBF kernel it's better to reduce the number of dimensions. You can put min_df = 10, max_features = 500 and consider a sample size of 40k points.
3. **Hyper paramter tuning (find best alpha in range [10^-4 to 10^4], and the best penalty among 'l1', 'l2')**
   - 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
4. **Feature importance**
   - When you are working on the linear kernel with BOW or TFIDF please print the top 10 best features for each of the positive and negative classes.
5. **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.
6. **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.

   - **Conclusion** (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 SVM

[5.1] Linear SVM

[5.1.1] Applying Linear SVM on BOW, SET 1

Applying Stochaistic Gradient Descent Algorith with a Hinge Lowss is works similar to LinearSVM model

And SgdClassifier is also cost effective than LinearSVM.

Mathematical formulation behind the Linearsvm is, Arg

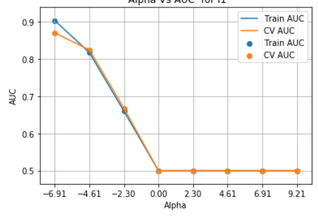
In [50]:
```
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import cross_val_score
from sklearn.metrics import roc_curve, auc, roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import SGDClassifier
```

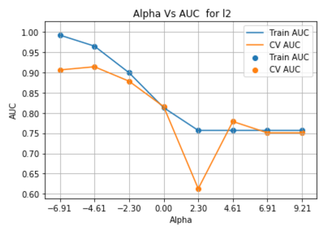
In [2]:
```
lrn_rt = [10**i for i in range(-4,5)]
Regularisation = ['l1','l2']
print("learning Rate  : ",lrn_rt)
print("Regularisation  : ",Regularisation)
```

```
learning Rate  :  [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]
Regularisation  :  ['l1', 'l2']
```

In [238]:
```
#set1_train_auc,set1_cv_auc
def plot_Auc_Lrn_Rt(Train_Auc,Cv_Auc,lrn_rt,Reg):
    plt.grid()
    plt.scatter(np.log(lrn_rt), Train_Auc, label='Train AUC')
    plt.plot(np.log(lrn_rt), Train_Auc, label='Train AUC')
    plt.scatter(np.log(lrn_rt), Cv_Auc, label='CV AUC')
    plt.plot(np.log(lrn_rt), Cv_Auc, label='CV AUC')
    plt.legend()
    plt.xticks(np.log(lrn_rt))
    plt.xlabel("Alpha")
    plt.ylabel("AUC")
    title = " Alpha Vs AUC  for " + Reg
    plt.title(title)
    plt.show()
```

In [237]:
```
#https://github.com/scikit-learn/scikit-learn/blob/master/doc/modules/sgd.rst

Set1_Weights_Cv   = []

for Reg in Regularisation:
    Set1_Train_Auc = []
    Set1_Cv_Auc       = []
    for i in lrn_rt:
        Sgdc = SGDClassifier(loss='hinge',penalty=Reg,n_iter=50, alpha=i,class_weight = "balanced",random_state=0)
        scores = cross_val_score(Sgdc, X_Bow_Cv, Y_cv, cv=10,  scoring='roc_auc')
        Cv_Auc = np.mean(scores)

        Sgdc.fit(X_Bow_Tr,Y_tr)
        predict_proba = Sgdc.decision_function(X_Bow_Tr)
        Train_Auc     = roc_auc_score(Y_tr, predict_proba)

        Set1_Train_Auc.append(Train_Auc)
        Set1_Cv_Auc.append(Cv_Auc)
    plot_Auc_Lrn_Rt(Set1_Train_Auc,Set1_Cv_Auc,lrn_rt,Reg)
```





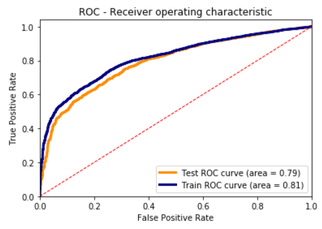
In [47]:
```
Set1_Opt_Lr_Rt = lrn_rt[1]
Set1_Opt_Reg = Regularisation[0]
```

In [51]:
```
Set1_Weights_L1 = []
Sgdc = SGDClassifier(loss='hinge',alpha =Set1_Opt_Reg,n_iter=50, penalty=Set1_Opt_Lr_Rt,class_weight="balanced",random_state=0)
Sgdc.fit(X_Bow_Tr,Y_tr)

Set1_Weights = Sgdc.coef_[0].tolist()

Set1_Tr_prob    =  Sgdc.decision_function(X_Bow_Tr)    # Probablity of TRAIN-Validation
Set1_Tst_prob   =  Sgdc.decision_function(X_Bow_Test)    # Probablity of Cross-Validation
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-51-538cb2be6bbb> in <module>()
      1 Set1_Weights_L1 = []
----> 2 Sgdc = SGDClassifier(loss='hinge',alpha =Set1_Opt_Reg,n_iter=50, penalty=Set1_Opt_Lr_Rt,class_weight="balanced",random_state=0)
      3 Sgdc.fit(X_Bow_Tr,Y_tr)
      4
      5 Set1_Weights = Sgdc.coef_[0].tolist()

~\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py in __init__(self, loss, penalty, alpha, l1_ratio, fit_intercept, max_iter, tol, shuffle, verbose, epsilon, n_jobs, random_state, learning_rate, eta0, power_t, class_weight, warm_start, average, n_iter)
    787             random_state=random_state, learning_rate=learning_rate, eta0=eta0,
    788             power_t=power_t, class_weight=class_weight, warm_start=warm_start,
--> 789             average=average, n_iter=n_iter)
    790
    791     def _check_proba(self):

~\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py in __init__(self, loss, penalty, alpha, l1_ratio, fit_intercept, max_iter, tol, shuffle, verbose, epsilon, n_jobs, random_state, learning_rate, eta0, power_t, class_weight, warm_start, average, n_iter)
    355                                           warm_start=warm_start,
    356                                           average=average,
--> 357                                           n_iter=n_iter)
    358         self.class_weight = class_weight
    359         self.n_jobs = int(n_jobs)

~\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py in __init__(self, loss, penalty, alpha, C, l1_ratio, fit_intercept, max_iter, tol, shuffle, verbose, epsilon, random_state, learning_rate, eta0, power_t, warm_start, average, n_iter)
     72         # current tests expect init to do parameter validation
     73         # but we are not allowed to set attributes
---> 74         self._validate_params(set_max_iter=False)
     75
     76     def set_params(self, *args, **kwargs):

~\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py in _validate_params(self, set_max_iter)
     91         if not (0.0 <= self.l1_ratio <= 1.0):
     92             raise ValueError("l1_ratio must be in [0, 1]")
---> 93         if self.alpha < 0.0:
     94             raise ValueError("alpha must be >= 0")
     95         if self.learning_rate in ("constant", "invscaling"):

TypeError: '<' not supported between instances of 'str' and 'float'
```

In [241]:
```
#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set1_tst_fpr, set1_tst_tpr, thresholds = roc_curve(Y_test,Set1_Tst_prob)
set1_tst_roc_auc = auc(set1_tst_fpr, set1_tst_tpr)


set1_train_fpr, set1_train_tpr, thresholds = roc_curve(Y_tr,Set1_Tr_prob)
set1_train_roc_auc = auc(set1_train_fpr, set1_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set1_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set1_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set1_tst_fpr, set1_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set1_tst_roc_auc)
plt.plot(set1_train_fpr, set1_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set1_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=1w, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
 Train Data      AUC for the Best Alpha   0.8130022108917986
 Test Validaton  AUC for the Best Alpha  0.7911610170132898
```

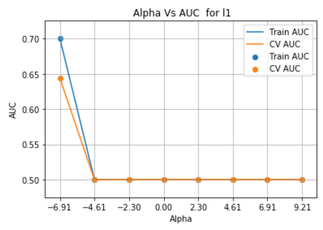


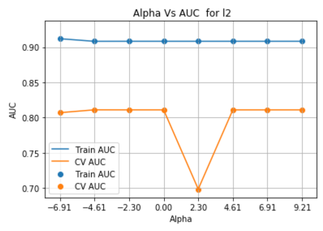
In [242]:
```
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    # it is recall(tpr) * precision((1-fpr))

    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return t,predictions
```

In [243]:
```
# Please write all the code with proper documentation
```

[5.1.2] Applying Linear SVM on TFIDF, SET 2

In [244]:
```
#https://github.com/scikit-learn/scikit-learn/blob/master/doc/modules/sgd.rst
Set2_Weights_Cv   = []
for Reg in Regularisation:
    Set2_Train_Auc  = []
    Set2_Cv_Auc     = []
    for i in lrn_rt:
        Sgdc = SGDClassifier(loss='hinge',penalty=Reg,n_iter=50, alpha=i,class_weight = "balanced",random_state=0)
        scores = cross_val_score(Sgdc, X_Tfidf_Cv, Y_cv, cv=10,  scoring='roc_auc')
        Cv_Auc = np.mean(scores)

        Sgdc.fit(X_Tfidf_Tr,Y_tr)
        predict_proba = Sgdc.decision_function(X_Tfidf_Tr)
        Train_Auc     = roc_auc_score(Y_tr, predict_proba)

        Set2_Train_Auc.append(Train_Auc)
        Set2_Cv_Auc.append(Cv_Auc)
    plot_Auc_Lrn_Rt(Set2_Train_Auc,Set2_Cv_Auc,lrn_rt,Reg)
```



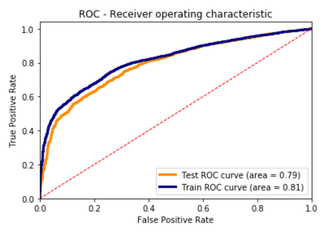


In [245]:
```
Set2_Opt_Lr_Rt = lrn_rt[1]
Set2_Opt_Reg = Regularisation[0]
```

In [246]:
```
Set2_Weights_L1 = []
Sgdc = SGDClassifier(loss='hinge',penalty=Set2_Opt_Reg,n_iter=50, alpha=Set2_Opt_Lr_Rt,class_weight="balanced",random_state=0)
Sgdc.fit(X_Bow_Tr,Y_tr)

Set2_Weights = Sgdc.coef_[0].tolist()

Set2_Tr_prob    =  Sgdc.decision_function(X_Bow_Tr)    # Probablity of TRAIN-Validation
Set2_Tst_prob   =  Sgdc.decision_function(X_Bow_Test)    # Probablity of Cross-Validation
```

In [247]:
```
#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set2_tst_fpr, set2_tst_tpr, thresholds = roc_curve(Y_test,Set2_Tst_prob)
set2_tst_roc_auc = auc(set2_tst_fpr, set2_tst_tpr)


set2_train_fpr, set2_train_tpr, thresholds = roc_curve(Y_tr,Set2_Tr_prob)
set2_train_roc_auc = auc(set2_train_fpr, set2_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set2_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set2_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set1_tst_fpr, set1_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set2_tst_roc_auc)
plt.plot(set1_train_fpr, set1_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set2_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

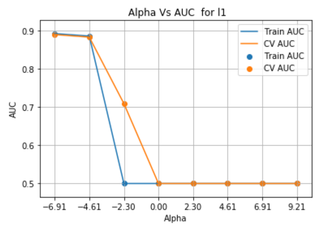
```
 Train Data      AUC for the Best Alpha   0.8130022108917986
 Test Validaton  AUC for the Best Alpha  0.7911610170132898
```

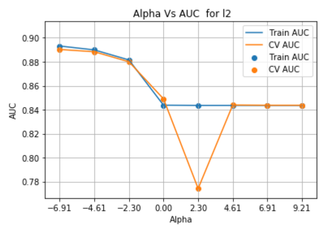


In [248]:
```
# Please write all the code with proper documentation
```

[5.1.3] Applying Linear SVM on AVG W2V, SET 3

In [249]:
```
Set3_Weights_Cv   = []
for Reg in Regularisation:
    Set3_Train_Auc  = []
    Set3_Cv_Auc     = []
    for i in lrn_rt:
        Sgdc = SGDClassifier(loss='hinge',penalty=Reg,n_iter=50, alpha=i,class_weight = "balanced",random_state=0)
        scores = cross_val_score(Sgdc, X_AvgW2V_Cv, Y_cv, cv=10,  scoring='roc_auc')
        Cv_Auc = np.mean(scores)

        Sgdc.fit(X_AvgW2V_Tr,Y_tr)
        predict_proba = Sgdc.decision_function(X_AvgW2V_Tr)
        Train_Auc     = roc_auc_score(Y_tr, predict_proba)

        Set3_Train_Auc.append(Train_Auc)
        Set3_Cv_Auc.append(Cv_Auc)
    plot_Auc_Lrn_Rt(Set3_Train_Auc,Set3_Cv_Auc,lrn_rt,Reg)
```



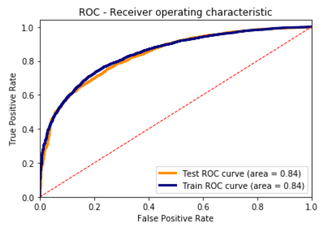


In [250]:
```
Set3_Opt_Lr_Rt = lrn_rt[1]
Set3_Opt_Reg = Regularisation[1]
```

In [251]:
```
Set3_Weights_L1 = []
Sgdc = SGDClassifier(loss='hinge',penalty=Set3_Opt_Reg,n_iter=50, alpha=Set3_Opt_Lr_Rt,class_weight="balanced",random_state=0)
Sgdc.fit(X_AvgW2V_Tr,Y_tr)

Set3_Weights = Sgdc.coef_[0].tolist()

Set3_Tr_prob    =  Sgdc.decision_function(X_AvgW2V_Tr)    # Probablity of TRAIN-Validation
Set3_Tst_prob   =  Sgdc.decision_function(X_AvgW2V_Test)    # Probablity of Cross-Validation
```

In [252]:
```
#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set3_tst_fpr, set3_tst_tpr, thresholds = roc_curve(Y_test,Set3_Tst_prob)
set3_tst_roc_auc = auc(set3_tst_fpr, set3_tst_tpr)


set3_train_fpr, set3_train_tpr, thresholds = roc_curve(Y_tr,Set3_Tr_prob)
set3_train_roc_auc = auc(set3_train_fpr, set3_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set3_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set3_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set3_tst_fpr, set3_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set3_tst_roc_auc)
plt.plot(set3_train_fpr, set3_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set3_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

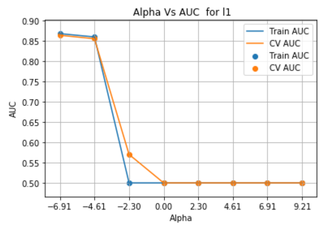
```
 Train Data      AUC for the Best Alpha   0.8435966436756972
 Test Validaton  AUC for the Best Alpha  0.8363840231737959
```

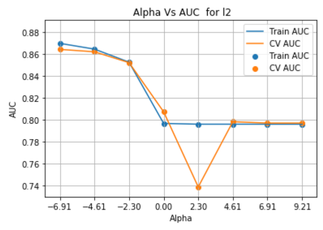


In [253]:
```
# Please write all the code with proper documentation
```

[5.1.4] Applying Linear SVM on TFIDF W2V, SET 4

In [254]:
```
Set4_Weights_Cv   = []
for Reg in Regularisation:
    Set4_Train_Auc  = []
    Set4_Cv_Auc     = []
    for i in lrn_rt:
        Sgdc = SGDClassifier(loss='hinge',penalty=Reg,n_iter=50, alpha=i,class_weight = "balanced",random_state=0)
        scores = cross_val_score(Sgdc, X_AvgW2VtfIdf_Cv, Y_cv, cv=10,  scoring='roc_auc')
        Cv_Auc = np.mean(scores)

        Sgdc.fit(X_AvgW2VtfIdf_Tr,Y_tr)
        predict_proba = Sgdc.decision_function(X_AvgW2VtfIdf_Tr)
        Train_Auc     = roc_auc_score(Y_tr, predict_proba)

        Set4_Train_Auc.append(Train_Auc)
        Set4_Cv_Auc.append(Cv_Auc)
    plot_Auc_Lrn_Rt(Set4_Train_Auc,Set4_Cv_Auc,lrn_rt,Reg)
```



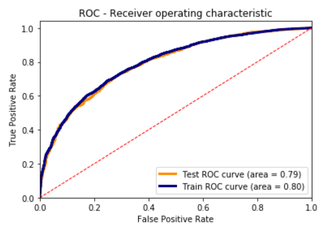


In [46]:
```
Set4_Opt_Lr_Rt = lrn_rt[1]
Set4_Opt_Reg = Regularisation[1]
```

In [256]:
```
Set4_Weights_L1 = []
Sgdc = SGDClassifier(loss='hinge',penalty=Set4_Opt_Reg,n_iter=50, alpha=Set4_Opt_Lr_Rt,class_weight="balanced",random_state=0)
Sgdc.fit(X_AvgW2VtfIdf_Tr,Y_tr)

Set4_Weights = Sgdc.coef_[0].tolist()

Set4_Tr_prob    =  Sgdc.decision_function(X_AvgW2VtfIdf_Tr)    # Probablity of TRAIN-Validation
Set4_Tst_prob   =  Sgdc.decision_function(X_AvgW2VtfIdf_Test)    # Probablity of Cross-Validation
```

In [257]:
```
#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set4_tst_fpr, set4_tst_tpr, thresholds = roc_curve(Y_test,Set4_Tst_prob)
set4_tst_roc_auc = auc(set4_tst_fpr, set4_tst_tpr)


set4_train_fpr, set4_train_tpr, thresholds = roc_curve(Y_tr,Set4_Tr_prob)
set4_train_roc_auc = auc(set4_train_fpr, set4_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set4_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set4_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set4_tst_fpr, set4_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set4_tst_roc_auc)
plt.plot(set4_train_fpr, set4_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set4_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
Train Data      AUC for the Best Alpha   0.7959121983587385
Test Validaton  AUC for the Best Alpha  0.7913294092271365
```



[5.2] RBF SVM

[5.2.1] Applying RBF SVM on BOW, SET 1

In [52]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
```

In [55]:
```
Penalty  = lrn_rt
Reg = "Rbf Kernel"
```

In [ ]:
```
#https://www.kaggle.com/azzion/svm-for-beginners-tutorial
Set5_Weights_Cv = []
Set5_Train_Auc  = []
Set5_Cv_Auc     = []
for i in Penalty:
    svc=SVC(kernel='rbf',C= i,class_weight = "balanced", random_state = 0,probability=True)
    scores = cross_val_score(svc, X_Bow_Cv, Y_cv, cv=10,  scoring='roc_auc')
    Cv_Auc = np.mean(scores)

    svc.fit(X_Bow_Tr,Y_tr)
    predict_proba = svc.predict_proba(X_Bow_Tr)
    Train_Auc     = roc_auc_score(Y_tr, predict_proba[:,1])

    Set5_Train_Auc.append(Train_Auc)
    Set5_Cv_Auc.append(Cv_Auc)
plot_Auc_Lrn_Rt(Set5_Train_Auc,Set5_Cv_Auc,Penalty,Reg)
```

In [ ]:
```
Set5_Opt_Lr_Rt = Penalty[3]
```

In [ ]:
```
Set5_Weights_L1 = []
svc=SVC(kernel='rbf',C= Set5_Opt_Lr_Rt,class_weight = "balanced", random_state = 0,probability=True)
svc.fit(X_Bow_Tr,Y_tr)

Set5_Tr_prob = svc.predict_proba(X_Bow_Tr)
Set5_Tst_prob = svc.predict_proba(X_Bow_Test)

#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set5_tst_fpr, set5_tst_tpr, thresholds = roc_curve(Y_test,Set5_Tst_prob[:,1])
set5_tst_roc_auc = auc(set5_tst_fpr, set5_tst_tpr)


set5_train_fpr, set5_train_tpr, thresholds = roc_curve(Y_tr,Set5_Tr_prob[:,1])
set5_train_roc_auc = auc(set5_train_fpr, set5_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set5_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set5_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set5_tst_fpr, set5_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set5_tst_roc_auc)
plt.plot(set5_train_fpr, set5_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set5_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

[5.2.2] Applying RBF SVM on TFIDF, SET 2

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

In [ ]:
```
#https://www.kaggle.com/azzion/svm-for-beginners-tutorial
Set6_Weights_Cv = []
Set6_Train_Auc  = []
Set6_Cv_Auc     = []
for i in Penalty:
    svc=SVC(kernel='rbf',C= i,class_weight = "balanced", random_state = 0,probability=True)
    scores = cross_val_score(svc, X_Tfidf_Cv, Y_cv, cv=10,  scoring='roc_auc')
    Cv_Auc = np.mean(scores)

    svc.fit(X_Tfidf_Tr,Y_tr)
    predict_proba = svc.predict_proba(X_Tfidf_Tr)
    Train_Auc     = roc_auc_score(Y_tr, predict_proba[:,1])

    Set6_Train_Auc.append(Train_Auc)
    Set6_Cv_Auc.append(Cv_Auc)
plot_Auc_Lrn_Rt(Set6_Train_Auc,Set6_Cv_Auc,Penalty,Reg)
```

In [ ]:
```
Set6_Opt_Lr_Rt = Penalty[6]
```

In [ ]:
```
Set6_Weights_L1 = []
svc=SVC(kernel='rbf',C= Set6_Opt_Lr_Rt,class_weight = "balanced", random_state = 0,probability=True)
svc.fit(X_Tfidf_Tr,Y_tr)

Set6_Tr_prob = svc.predict_proba(X_Tfidf_Tr)
Set6_Tst_prob = svc.predict_proba(X_Tfidf_Test)

#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set6_tst_fpr, set6_tst_tpr, thresholds = roc_curve(Y_test,Set6_Tst_prob[:,1])
set6_tst_roc_auc = auc(set6_tst_fpr, set6_tst_tpr)


set6_train_fpr, set6_train_tpr, thresholds = roc_curve(Y_tr,Set6_Tr_prob[:,1])
set6_train_roc_auc = auc(set6_train_fpr, set6_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set6_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set6_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set6_tst_fpr, set6_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set6_tst_roc_auc)
plt.plot(set6_train_fpr, set6_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set6_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

[5.2.3] Applying RBF SVM on AVG W2V, SET 3

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

In [ ]:
```
#https://www.kaggle.com/azzion/svm-for-beginners-tutorial
Set7_Weights_Cv = []
Set7_Train_Auc  = []
Set7_Cv_Auc     = []
for i in Penalty:
    svc=SVC(kernel='rbf',C= i,class_weight = "balanced", random_state = 0,probability=True)
    scores = cross_val_score(svc, X_AvgW2V_Cv, Y_cv, cv=10,  scoring='roc_auc')
    Cv_Auc = np.mean(scores)

    svc.fit(X_AvgW2V_Tr,Y_tr)
    predict_proba = svc.predict_proba(X_AvgW2V_Tr)
    Train_Auc     = roc_auc_score(Y_tr, predict_proba[:,1])

    Set7_Train_Auc.append(Train_Auc)
    Set7_Cv_Auc.append(Cv_Auc)
plot_Auc_Lrn_Rt(Set7_Train_Auc,Set7_Cv_Auc,Penalty,Reg)
```

In [ ]:
```
Set7_Opt_Lr_Rt = Penalty[6]
```

In [ ]:
```
Set7_Weights_L1 = []
svc=SVC(kernel='rbf',C= Set7_Opt_Lr_Rt,class_weight = "balanced", random_state = 0,probability=True)
svc.fit(X_AvgW2V_Tr,Y_tr)

Set7_Tr_prob = svc.predict_proba(X_AvgW2V_Tr)
Set7_Tst_prob = svc.predict_proba(X_AvgW2V_Test)

#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set7_tst_fpr, set7_tst_tpr, thresholds = roc_curve(Y_test,Set7_Tst_prob[:,1])
set7_tst_roc_auc = auc(set7_tst_fpr, set7_tst_tpr)


set7_train_fpr, set7_train_tpr, thresholds = roc_curve(Y_tr,Set7_Tr_prob[:,1])
set7_train_roc_auc = auc(set7_train_fpr, set7_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set7_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set7_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set7_tst_fpr, set7_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set7_tst_roc_auc)
plt.plot(set7_train_fpr, set7_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set7_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

[5.2.4] Applying RBF SVM on TFIDF W2V, SET 4

In [ ]:
```
# Please write all the code with proper documentation
#https://www.kaggle.com/azzion/svm-for-beginners-tutorial
Set8_Weights_Cv = []
Set8_Train_Auc  = []
Set8_Cv_Auc     = []
for i in Penalty:
    svc=SVC(kernel='rbf',C= i,class_weight = "balanced", random_state = 0,probability=True)
    scores = cross_val_score(svc, X_AvgW2VtfIdf_Cv, Y_cv, cv=10,  scoring='roc_auc')
    Cv_Auc = np.mean(scores)

    svc.fit(X_AvgW2VtfIdf_Tr,Y_tr)
    predict_proba = svc.predict_proba(X_AvgW2VtfIdf_Tr)
    Train_Auc     = roc_auc_score(Y_tr, predict_proba[:,1])

    Set8_Train_Auc.append(Train_Auc)
    Set8_Cv_Auc.append(Cv_Auc)
plot_Auc_Lrn_Rt(Set8_Train_Auc,Set8_Cv_Auc,Penalty,Reg)
```

In [ ]:
```
Set8_Opt_Lr_Rt = Penalty[6]
```

In [ ]:
```
Set8_Weights_L1 = []
svc=SVC(kernel='rbf',C= Set8_Opt_Lr_Rt,class_weight = "balanced", random_state = 0,probability=True)
svc.fit(X_Tfidf_Tr,Y_tr)

Set8_Tr_prob = svc.predict_proba(X_Tfidf_Tr)
Set8_Tst_prob = svc.predict_proba(X_Tfidf_Test)

#https://qiita.com/bmj0114/items/460424c110a8ce22d945
set8_tst_fpr, set8_tst_tpr, thresholds = roc_curve(Y_test,Set8_Tst_prob[:,1])
set8_tst_roc_auc = auc(set8_tst_fpr, set8_tst_tpr)


set8_train_fpr, set8_train_tpr, thresholds = roc_curve(Y_tr,Set8_Tr_prob[:,1])
set8_train_roc_auc = auc(set8_train_fpr, set8_train_tpr)


print(" Train Data      AUC for the Best Alpha  ", set8_train_roc_auc)
print(" Test Validaton  AUC for the Best Alpha ",set8_tst_roc_auc)

lw=1
plt.figure()
plt.plot(set8_tst_fpr, set8_tst_tpr, color='darkorange', lw=3, label='Test ROC curve (area = %0.2f)' % set8_tst_roc_auc)
plt.plot(set8_train_fpr, set8_train_tpr, color='navy', lw=3, label='Train ROC curve (area = %0.2f)' % set8_train_roc_auc)
plt.plot([0, 1], [0,1], color='red', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.04])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC - Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

In [ ]:
```
#https://www.kaggle.com/azzion/svm-for-beginners-tutorial
param_grid = {'C': [1,5,7,10,15,25,50],
              'gamma': [0.1,0.5,0.10,0.25,0.50,1,2]}
param_grid = {'C': [0.1,1],
              'gamma': [0.1,0.5]}
GS = GridSearchCV(SVC(kernel='rbf'),param_grid,cv=10,scoring='roc_auc')
#,X_AvgW2VtfIdf_Tr , X_AvgW2VtfIdf_Cv
GS.fit(X_AvgW2VtfIdf_Cv,Y_cv)
```

[6] Conclusions

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