Amazon Fine Food Reviews Analysis

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

EDA: 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).

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

[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 [2]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```python
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
!pip install pandasql
import pandasql as ps
!pip install kaggle
```

```
Collecting pandasql
  Downloading https://files.pythonhosted.org/packages/6b/c4/ee4096ffa2e
eeca0c749b26f0371bd26aa5c8b611c43de99a4f86d3de0a7/pandasql-0.7.3.tar.gz
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-p
ackages (from pandasql) (1.14.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-
packages (from pandasql) (0.22.0)
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.6/d
ist-packages (from pandasql) (1.3.1)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/
dist-packages (from pandas->pandasql) (2018.9)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/pyt
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
hon3.6/dist-packages (from pandas->pandasql) (2.5.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dis
t-packages (from python-dateutil>=2->pandas->pandasql) (1.11.0)
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/53/6c/18/b87a2e5fa8a82e9
c026311de56210b8d1c01846e18a9607fc9
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-
packages (1.5.3)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/
python3.6/dist-packages (from kaggle) (1.22)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/di
st-packages (from kaggle) (1.11.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist
-packages (from kaggle) (2019.3.9)
Requirement already satisfied: python-dateutil in /usr/local/lib/python
3.6/dist-packages (from kaggle) (2.5.3)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dis
t-packages (from kaggle) (2.18.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-pa
ckages (from kaggle) (4.28.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python
3.6/dist-packages (from kaggle) (3.0.1)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/
python3.6/dist-packages (from requests->kaggle) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python
3.6/dist-packages (from requests->kaggle) (2.6)
Requirement already satisfied: text-unidecode==1.2 in /usr/local/lib/py
thon3.6/dist-packages (from python-slugify->kaggle) (1.2)
```

In [0]:

```
#upload the credentials of the kaggle account
from google.colab import files
files.upload()
```

In [0]:

```
#before importing the dataset we want to use this code
# The Kaggle API client expects this file to be in ~/.kaggle,
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

# This permissions change avoids a warning on Kaggle tool startup.
!chmod 600 ~/.kaggle/kaggle.json
```

In [0]:
```
!ls
```

In [0]:
```
#import the Amazon fine food review dataset from kaggle
!kaggle datasets download -d snap/amazon-fine-food-reviews --force
```

In [0]:
```
!unzip amazon-fine-food-reviews.zip
```

In [56]:
```
# 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 50
0000 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 Sco
re != 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 100000""", con)

# Give reviews with Score>3 a positive rating(1), and reviews with a sc
ore<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
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
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 (100000, 10)
```

Out[56]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulnes
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 [0]:

```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [58]:
```
print(display.shape)
display.head()
```

(80668, 7)

Out[58]:

	UserId	ProductId	ProfileName	Time	Score	Text	COU
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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	UserId	ProductId	ProfileName	Time	Score	Text	COU
4	#oc-R12KPBODL2B5ZD	B007OSBE1U	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

```
In [59]: display[display['UserId']=='AZY10LLTJ71NX']
```

Out[59]:

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

```
In [60]: display['COUNT(*)'].sum()
```

Out[60]: 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 [61]: display= pd.read_sql_query("""
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[61]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln
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

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln
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 [0]:

```
#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 [63]:

```
#Deduplication of entries
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time"
,"Text"}, keep='first', inplace=False)
final.shape
```

Out[63]: (87775, 10)

In [64]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[64]: 87.775

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 [65]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[65]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	1

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln
1	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	2

In [0]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [67]:
```
#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()
```

```
(87773, 10)
```

Out[67]:
```
1    73592
0    14181
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:-

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
t'
# <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', 'o
urs', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselve
s', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
s', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'th
is', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'h
ave', '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', 'h
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
ow', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 's
o', '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', "is
n'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 [0]:

```
# https://stackoverflow.com/a/47091490/4084039
# Method will replace Contracted words to normal words.
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 [70]:

```
# Combining all the above stundents
from bs4 import BeautifulSoup
from tqdm import tqdm

preprocessed_reviews = []
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# 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%|██████████| 87773/87773 [00:38<00:00, 2302.57it/s]
```

In [71]:

```
#Printing one random review:
preprocessed_reviews[2134]
```

Out[71]:

```
'strange taste kind like relish jelly not good toast really good ham ch
eese melt'
```

[3.2]. Preprocessing Review summary

In [72]:

```
#Sincw URL is very less in the summary so I am directly applying all th
e stundants at a time.

from tqdm import tqdm
preprocessed_Summary = []
# tqdm is for printing the status bar
for Summary in tqdm(final['Summary'].values):
    Summary = re.sub(r"http\S+", "", Summary)
    Summary = BeautifulSoup(Summary, 'lxml').get_text()
    #Summary = remove_tags(Summary)
    Summary = decontracted(Summary)
    Summary = re.sub("\S*\d\S*", "", Summary).strip()
    Summary = re.sub('[^A-Za-z]+', ' ', Summary)
    # https://gist.github.com/sebleier/554280
    Summary = ' '.join(e.lower() for e in Summary.split() if e.lower()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
not in stopwords)
    preprocessed_Summary.append(Summary.strip())
```

```
100%|██████████| 87773/87773 [00:27<00:00, 3247.28it/s]
```

In [73]:
```
#printing one random summary:
preprocessed_Summary[24323]
```

Out[73]:
```
'home owner'
```

[4] Feature Engineering

In [0]:
```
#Merging the Review text and review summary together.

final["Clean_Text"] =preprocessed_reviews
final["Clean_Summary"] =preprocessed_Summary
# Combining them together in Final_Text Field.
final["Final_Text"] =final['Clean_Text'].values+" "+final['Clean_Summar
y'].values
```

In [75]:
```
final.head()
```

Out[75]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He
22620	24750	2734888454	A13ISQV0U9GZIC	Sandikaye	1	1

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He
22621	24751	2734888454	A1C298ITT645B6	Hugh G. Pritchard	0	0
70677	76870	B00002N8SM	A19Q006CSFT011	Arlielle	0	0
70676	76869	B00002N8SM	A1FYH4S02BW7FN	wonderer	0	0
70675	76868	B00002N8SM	AUE8TB5VHS6ZV	eyeofthestorm	0	0

In [76]:
```
#printing one data from final text column
final.iloc[2134].Final_Text
```

Out[76]:
```
'strange taste kind like relish jelly not good toast really good ham ch
eese melt strange good'
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

[5] Spliting into Train and test

In [35]:
```
# Some more imports
!pip install -U scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection
```

```
Requirement already up-to-date: scikit-learn in /usr/local/lib/python3.
6/dist-packages (0.20.3)
Requirement already satisfied, skipping upgrade: numpy>=1.8.2 in /usr/l
ocal/lib/python3.6/dist-packages (from scikit-learn) (1.14.6)
Requirement already satisfied, skipping upgrade: scipy>=0.13.3 in /usr/
local/lib/python3.6/dist-packages (from scikit-learn) (1.1.0)
```

In [77]:
```
#Keeping the Final_Data in a temp variable
temp = final
temp.iloc[2134].Final_Text
```

Out[77]:
```
'strange taste kind like relish jelly not good toast really good ham ch
eese melt strange good'
```

In [0]:
```
#Use to revert back the Final_Data dataset
final = temp
```

In [78]:
```
final.head(5)
```

Out[78]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He
22620	24750	2734888454	A13ISQV0U9GZIC	Sandikaye	1	1
22621	24751	2734888454	A1C298ITT645B6	Hugh G. Pritchard	0	0
70677	76870	B00002N8SM	A19Q006CSFT011	Arlielle	0	0
70676	76869	B00002N8SM	A1FYH4S02BW7FN	wonderer	0	0
70675	76868	B00002N8SM	AUE8TB5VHS6ZV	eyeofthestorm	0	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [0]:
```python
# Sorting data based on time
#final_Data["Time"] = pd.to_datetime(final_Data["Time"], unit = "s")
#final_Data = final_Data.sort_values(by = "Time")

final = final.sort_values('Time', axis=0, ascending=True, inplace=False
, kind='quicksort', na_position='last')
```

In [40]:
```python
final.head(10)
```

Out[40]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNu
70688	76882	B00002N8SM	A32DW342WBJ6BX	Buttersugar	0
1146	1245	B00002Z754	A29Z5PI9BW2PU3	Robbie	7
1145	1244	B00002Z754	A3B8RCEI0FXFI6	B G Chase	10
28086	30629	B00008RCMI	A19E94CF5O1LY7	Andrew Arnold	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessN
28087	30630	B00008RCMI	A284C7M23F0APC	A. Mendoza	0
61299	66610	B0000SY9U4	A3EEDHNHI4WNSH	Joanna J. Young	23
38740	42069	B0000EIEQU	A1YMJX4YWCE6P4	Jim Carson "http://www.jimcarson.com"	12
38889	42227	B0000A0BS8	A1IU7S4HCK1XK0	Joanna Daneman	5
38888	42226	B0000A0BS8	A23GFTVIETX7DS	Debbie Lee Wesselmann	5
10992	11991	B0000T15M8	A2928LJN5IISB4	chatchi	5

In [80]:
```
final.tail(60)
```

Out[80]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
87501	95241	B00401OZ1U	A1KIL93AY6MFGS	John K. Kirk	0
63160	68621	B005IOXBY0	A1ORVAUR5C5N8X	amondigirl	0
66057	71787	B007RTR8AC	A2PZM8DT1KGT10	Edwina E. Cowgill "book lover"	0
41621	45226	B00443Z35G	A3G23SMM1E1KPV	L. E. Scott	0
9513	10404	B005HI55CS	A36ERNIM0TKG3T	Donald E. Bolton	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
30235	32932	B001P05K8Q	A3L0B5NBTQ7ZHO	Julie	0
96778	105165	B005EF0HZ4	A2A5Z7LC91EFVA	Gretchen Casey	0
96779	105166	B005EF0HZ4	A1JXSMYVHFPWM1	marsha m beers	0
32585	35471	B000WSHV1Q	A1YT628H711FN7	Laura Tomevi	0
96263	104607	B000FACFIA	A1TTGEM50SIS2R	Abby	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
75382	82028	B0019GZ7Z2	ANSBPV2CZVZ39	zena3546 "zena3546"	0
7451	8135	B0019GVYR2	ACSO5EDO1UMZ5	SeekingBodhi	0
69746	75882	B002W1F6TK	ANSBPV2CZVZ39	zena3546 "zena3546"	0
24496	26772	B004ZY4TK4	A4IL0CLL27Q33	D. Brennan	0
86066	93711	B001NZPFB0	A3318V6FJ2KIII	T. Dennis	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
75877	82567	B008FXKOI2	A3RKYD8IUC5S0N	Foureyedsnail	0
84914	92421	B007TGDXMK	A1BSGNCHEI1PJI	Kelly Bowser "Runs W Scissors"	0
5472	5924	B00523NRVO	A2JDXKFZ0PFHKU	James W. Shondel	0
38043	41317	B008NDSNAU	A2Z0XFW79HXASE	Kelby Scandrett "Kaiahso"	0
19181	20930	B001L1MKLY	A38XYFHXEUNUW6	bleaufire	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
85745	93367	B007TGDXMU	AAMUNRK134Y5P	Tony Schy	0
85744	93366	B007TGDXMU	AGUBQN9M09VQ4	Inger Jackson	0
85743	93365	B007TGDXMU	A9JMG87TELB6N	Ludmila Newman	0
55730	60463	B003QNJYXM	AYTSBGA5A3UWI	Imran Ali	0
55731	60464	B003QNJYXM	A2E2F8WSUB33VE	Maria A. Alfonzo	0
55158	59851	B001EQ5KTK	A3GS4GWPIBV0NT	R. Chester "ricki1966"	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
83618	91002	B001E5E470	AGADB4E6N1EDS	L. A. Stephenson "Mom of four"	0
8731	9564	B001EQ5IPQ	AA2104NO2VE8H	Lakshminarayan Iyer	0
78920	85816	B005XGH78E	A2YGWCOC3LM3KH	Luke R. McAllister	0
93098	101227	B006N0KV8W	A1OA7ZKRQH98C8	M Mills	0
42269	45991	B007VQQT1K	A34P4V70RNC2YV	S. Guss	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
76059	82772	B0049K99RW	A1Y73Y4VX3AJMZ	Rispir Chrone	0
25112	27424	B003WEFSAI	A37O0JPLJ8BOXP	Texaschick59	0
29158	31794	B0049D7HRS	A3LR9HCV3D96I3	Gypsy Healer	0
87843	95629	B000LKXDXU	A2J3PR6J36UTVH	Joyce	0
97624	106071	B007JTKEQK	A1DOMJI7GXGPNY	Jyouk	0
14526	15842	B007TJGZ5E	A3UOYYQS5Z47MS	David A. Levin "DaveL"	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
14300	15605	B000255OIG	AUINI96NMGXUI	Kkrys23	0
14299	15604	B000255OIG	A3SSEJ8IEM4YGW	Seagaul	0
82884	90215	B00866AM2G	ADTOX2JFWWA0B	Arnos Vale	0
82885	90216	B00866AM2G	AY839W9JQDZM2	Daniella	0
15069	16426	B007TJGZ54	A29BJSTYH9W3JI	Harry	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
43703	47562	B004M0Y8T8	A2QJS6MHTIFSRI	Georgie	0
13539	14784	B000S859NC	A2H7STZ2URUCOE	Christopher Whedon "the odd bead"	0
52220	56723	B0012XBD7I	A32NC2UF34RJQY	D. Pagliassotti	0
55100	59787	B002K9BG16	A30A7W9CZ77GFY	Cecelia Thomas "Lady Kinrowan"	0
89213	97089	B004O8KBK8	A1JPKFGGF128X1	MTNick	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
6548	7178	B004OQLIHK	AKHQMSUORSA91	Pen Name	0
60967	66252	B007OSBGOK	A10QOESY9VJ9K	Gina	0
43268	47077	B001C4PKIK	A3IMXYITIO8WHN	Thomas R. Jackson	0
16026	17512	B0045Z6K50	A3HM6TNYB7FNDL	C. Furman	0
90340	98294	B0002LY6W0	A1BX08Y0GIT5RU	L. Nguyen "Always on the lookout for a good d...	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
76594	83330	B005ZBZLT4	AAMUNRK134Y5P	Tony Schy	0
78715	85601	B003ZURM80	A1O6MADFNBRX7H	Denise Lake	0
50708	55049	B000IHJEDE	A2DFSA2JXQKVY3	C-Rush	0
76593	83329	B005ZBZLT4	A308RR8J9NJOOZ	Josh	0
22401	24518	B0016JJEFG	AO9WE22147CRH	Arvind Rajan	0
56673	61474	B005YVU4A6	A2LU545SISQOJ8	Kelly	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
37074	40274	B005VOOT52	A2FKFQQPU498JT	cc	0
5259	5703	B009WSNWC4	AMP7K1O84DH1T	ESTY	0

In [0]:
```
# split the data set into train and test
X_Train, X_Test, Y_Train, Y_Test = train_test_split(final, final['Scor
e'], test_size=0.3, random_state=42,shuffle = False)
```

In [82]:
```
print("Size of X_Test :",len(X_Test))
print("Size of Y_Test :",len(Y_Test))
print('='*50)
print("Size of X-Train :",len(X_Train))
print("Size of Y_Train :",len(Y_Train))
```

```
Size of X_Test : 26332
Size of Y_Test : 26332
==================================================
Size of X-Train : 61441
Size of Y_Train : 61441
```

In [83]:
```
X_Train.head()
```

Out[83]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Hel

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Hel
70688	76882	B00002N8SM	A32DW342WBJ6BX	Buttersugar	0	0
1146	1245	B00002Z754	A29Z5PI9BW2PU3	Robbie	7	7
1145	1244	B00002Z754	A3B8RCEI0FXFI6	B G Chase	10	10
28086	30629	B00008RCMI	A19E94CF5O1LY7	Andrew Arnold	0	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Hel
28087	30630	B00008RCMI	A284C7M23F0APC	A. Mendoza	0	0

In [84]:
```
X_Test.tail(60)
```

Out[84]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
87501	95241	B00401OZ1U	A1KIL93AY6MFGS	John K. Kirk	0
63160	68621	B005IOXBY0	A1ORVAUR5C5N8X	amondigirl	0
66057	71787	B007RTR8AC	A2PZM8DT1KGT10	Edwina E. Cowgill "book lover"	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
41621	45226	B00443Z35G	A3G23SMM1E1KPV	L. E. Scott	0
9513	10404	B005HI55CS	A36ERNIM0TKG3T	Donald E. Bolton	0
30235	32932	B001P05K8Q	A3L0B5NBTQ7ZHO	Julie	0
96778	105165	B005EF0HZ4	A2A5Z7LC91EFVA	Gretchen Casey	0
96779	105166	B005EF0HZ4	A1JXSMYVHFPWM1	marsha m beers	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
32585	35471	B000WSHV1Q	A1YT628H711FN7	Laura Tomevi	0
96263	104607	B000FACFIA	A1TTGEM50SIS2R	Abby	0
75382	82028	B0019GZ7Z2	ANSBPV2CZVZ39	zena3546 "zena3546"	0
7451	8135	B0019GVYR2	ACSO5EDO1UMZ5	SeekingBodhi	0
69746	75882	B002W1F6TK	ANSBPV2CZVZ39	zena3546 "zena3546"	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
24496	26772	B004ZY4TK4	A4IL0CLL27Q33	D. Brennan	0
86066	93711	B001NZPFB0	A3318V6FJ2KIII	T. Dennis	0
75877	82567	B008FXKOI2	A3RKYD8IUC5S0N	Foureyedsnail	0
84914	92421	B007TGDXMK	A1BSGNCHEI1PJI	Kelly Bowser "Runs W Scissors"	0
5472	5924	B00523NRVO	A2JDXKFZ0PFHKU	James W. Shondel	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
38043	41317	B008NDSNAU	A2Z0XFW79HXASE	Kelby Scandrett "Kaiahso"	0
19181	20930	B001L1MKLY	A38XYFHXEUNUW6	bleaufire	0
85745	93367	B007TGDXMU	AAMUNRK134Y5P	Tony Schy	0
85744	93366	B007TGDXMU	AGUBQN9M09VQ4	Inger Jackson	0
85743	93365	B007TGDXMU	A9JMG87TELB6N	Ludmila Newman	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
55730	60463	B003QNJYXM	AYTSBGA5A3UWI	Imran Ali	0
55731	60464	B003QNJYXM	A2E2F8WSUB33VE	Maria A. Alfonzo	0
55158	59851	B001EQ5KTK	A3GS4GWPIBV0NT	R. Chester "ricki1966"	0
83618	91002	B001E5E470	AGADB4E6N1EDS	L. A. Stephenson "Mom of four"	0
8731	9564	B001EQ5IPQ	AA2104NO2VE8H	Lakshminarayan Iyer	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
78920	85816	B005XGH78E	A2YGWCOC3LM3KH	Luke R. McAllister	0
93098	101227	B006N0KV8W	A1OA7ZKRQH98C8	M Mills	0
42269	45991	B007VQQT1K	A34P4V70RNC2YV	S. Guss	0
76059	82772	B0049K99RW	A1Y73Y4VX3AJMZ	Rispir Chrone	0
25112	27424	B003WEFSAI	A37O0JPLJ8BOXP	Texaschick59	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
29158	31794	B0049D7HRS	A3LR9HCV3D96I3	Gypsy Healer	0
87843	95629	B000LKXDXU	A2J3PR6J36UTVH	Joyce	0
97624	106071	B007JTKEQK	A1DOMJI7GXGPNY	Jyouk	0
14526	15842	B007TJGZ5E	A3UOYYQS5Z47MS	David A. Levin "DaveL"	0
14300	15605	B000255OIG	AUINI96NMGXUI	Kkrys23	0
14299	15604	B000255OIG	A3SSEJ8IEM4YGW	Seagaul	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
82884	90215	B00866AM2G	ADTOX2JFWWA0B	Arnos Vale	0
82885	90216	B00866AM2G	AY839W9JQDZM2	Daniella	0
15069	16426	B007TJGZ54	A29BJSTYH9W3JI	Harry	0
43703	47562	B004M0Y8T8	A2QJS6MHTIFSRI	Georgie	0
13539	14784	B000S859NC	A2H7STZ2URUCOE	Christopher Whedon "the odd bead"	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
52220	56723	B0012XBD7I	A32NC2UF34RJQY	D. Pagliassotti	0
55100	59787	B002K9BG16	A30A7W9CZ77GFY	Cecelia Thomas "Lady Kinrowan"	0
89213	97089	B004O8KBK8	A1JPKFGGF128X1	MTNick	0
6548	7178	B004OQLIHK	AKHQMSUORSA91	Pen Name	0
60967	66252	B007OSBGOK	A10QOESY9VJ9K	Gina	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
43268	47077	B001C4PKIK	A3IMXYITIO8WHN	Thomas R. Jackson	0
16026	17512	B0045Z6K50	A3HM6TNYB7FNDL	C. Furman	0
90340	98294	B0002LY6W0	A1BX08Y0GIT5RU	L. Nguyen "Always on the lookout for a good d...	0
76594	83330	B005ZBZLT4	AAMUNRK134Y5P	Tony Schy	0
78715	85601	B003ZURM80	A1O6MADFNBRX7H	Denise Lake	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerat
50708	55049	B000IHJEDE	A2DFSA2JXQKVY3	C-Rush	0
76593	83329	B005ZBZLT4	A308RR8J9NJOOZ	Josh	0
22401	24518	B0016JJEFG	AO9WE22147CRH	Arvind Rajan	0
56673	61474	B005YVU4A6	A2LU545SISQOJ8	Kelly	0
37074	40274	B005VOOT52	A2FKFQQPU498JT	cc	0
5259	5703	B009WSNWC4	AMP7K1O84DH1T	ESTY	0

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [0]:
```
X_Train = X_Train['Final_Text']
X_Test = X_Test['Final_Text']
```

Observation The Dataset is splitted properly based on time

[5] Featurization

[5.1] BAG OF WORDS

In [88]:
```
#BoW For Train
count_vect = CountVectorizer()

X_Train_BOW = count_vect.fit_transform(X_Train)

#BoW For Test
X_Test_BOW = count_vect.transform(X_Test)

print("the shape of out Train BOW vectorizer ",X_Train_BOW.get_shape())
print("the shape of out Test BOW vectorizer ",X_Test_BOW.get_shape())
```

```
the shape of out Train BOW vectorizer  (61441, 47483)
the shape of out Test BOW vectorizer  (26332, 47483)
```

[4.3] TF-IDF

In [89]:
```
#TF_IDF For Train
tf_idf_vect = TfidfVectorizer()

X_Train_TFIDF = tf_idf_vect.fit_transform(X_Train)

#TF_IDF For Test
X_Test_TFIDF = tf_idf_vect.transform(X_Test)
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print("the shape of out Train TFIDF vectorizer ",X_Train_TFIDF.get_shap
e())
print("the shape of out Test TFIDF vectorizer ",X_Test_TFIDF.get_shape
())
```

```
the shape of out Train TFIDF vectorizer  (61441, 47483)
the shape of out Test TFIDF vectorizer  (26332, 47483)
```

[5] Assignment 4: Apply Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
    - SET 1:Review text, preprocessed one converted into vectors using (BOW)
    - SET 2:Review text, preprocessed one converted into vectors using (TFIDF)
2. **The hyper paramter tuning(find best Alpha)**
    - Find the best hyper parameter which will give the maximum AUC value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning
3. **Feature importance**
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB and print their corresponding feature names
4. **Feature engineering**

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

5. **Representation of results**

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
- Along with plotting ROC curve, you need to print the [confusion matrix](#) with predicted and original labels of test data points. Please visualize your confusion matrices using [seaborn heatmaps.](#)

6. [**Conclusion**](#)

- [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 link](#)

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.](#)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Applying Multinomial Naive Bayes

[5.1] Applying Naive Bayes on BOW, SET 1

In [90]:
```
# Please write all the code with proper documentation

alpha= [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]

from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV

nb=MultinomialNB()
parameters= {'alpha':alpha}
clf=GridSearchCV(nb, parameters, cv=3, scoring='roc_auc')
clf.fit(X_Train_BOW, Y_Train)

train_auc=clf.cv_results_['mean_train_score']
train_auc_std=clf.cv_results_['std_train_score']
cv_auc=clf.cv_results_['mean_test_score']
cv_auc_std=clf.cv_results_['std_test_score']

plt.plot(alpha, train_auc, label='Train AUC')

plt.plot(alpha, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.xscale('log')
plt.title("ERROR PLOTS")
plt.show()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

In [91]:
```
print(clf.best_params_)
print(clf.best_estimator_)
print(clf.best_score_)
```

```
{'alpha': 0.1}
MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
0.9268449054725256
```

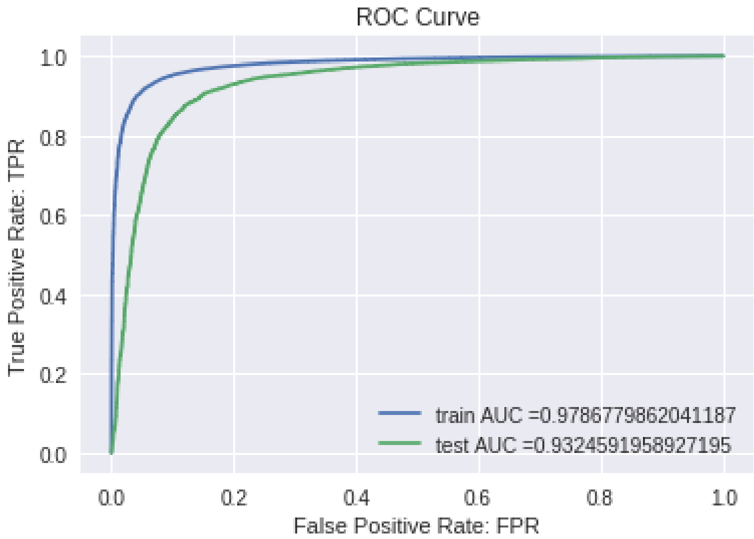
In [94]:
```
optimal_NB=MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
optimal_NB.fit(X_Train_BOW, Y_Train)

train_fpr, train_tpr, thresholds= roc_curve(Y_Train, optimal_NB.predict
_proba(X_Train_BOW)[:,1])
test_fpr, test_tpr, thresholds=roc_curve(Y_Test, optimal_NB.predict_pro
ba(X_Test_BOW)[:,1])
train_bow_acc =auc(train_fpr, train_tpr)
test_bow_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(train_bow_acc))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(test_bow_acc))
plt.legend()
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
plt.xlabel("False Positive Rate: FPR")
plt.ylabel("True Positive Rate: TPR")
plt.title("ROC Curve")
plt.show()
```



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

In [100]:

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

class_features=optimal_NB.feature_log_prob_
#  row_0 is for 'negative' class and row_1 is for 'positive' class
negative_features=class_features[0]
positive_features=class_features[1]

feature_names=count_vect.get_feature_names()

# Sorting 'positive_features' in descending order using argsort()functi
on. sorted_positive_features holds the index of top +ve features

sorted_positive_features=np.argsort(positive_features)[::-1]
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
print("\n\nTop 10 Important Features and their log probabilities For Po
sitive Class :\n\n")

for i in list(sorted_positive_features[0:10]):
  print("%s\t -->\t%f  "%(feature_names[i] , positive_features[i]))
```

```
Top 10 Important Features and their log probabilities For Positive Clas
s :


not       -->    -3.754726
great     -->    -4.435263
good      -->    -4.500713
like      -->    -4.562124
one       -->    -4.910012
love      -->    -4.956622
taste     -->    -4.966760
tea       -->    -4.972744
coffee    -->    -4.998359
flavor    -->    -5.067164
```

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

In [101]:

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

# Sorting 'positive_features' in descending order using argsort()functi
on. sorted_negative_features holds the index of top -ve features

sorted_negative_features=np.argsort(negative_features)[::-1]

print("\n\nTop 10 Important Features and their log probabilities For Ne
gative Class :\n\n")

for i in list(sorted_negative_features[0:10]):
  print("%s\t -->\t%f  "%(feature_names[i] , negative_features[i]))
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Top 10 Important Features and their log probabilities For Negative Clas
s :


not       -->    -3.215333
like      -->    -4.370493
taste     -->    -4.629433
would     -->    -4.679110
product   -->    -4.713654
one       -->    -4.897358
good      -->    -5.006622
coffee    -->    -5.071734
flavor    -->    -5.142142
no        -->    -5.144805
```

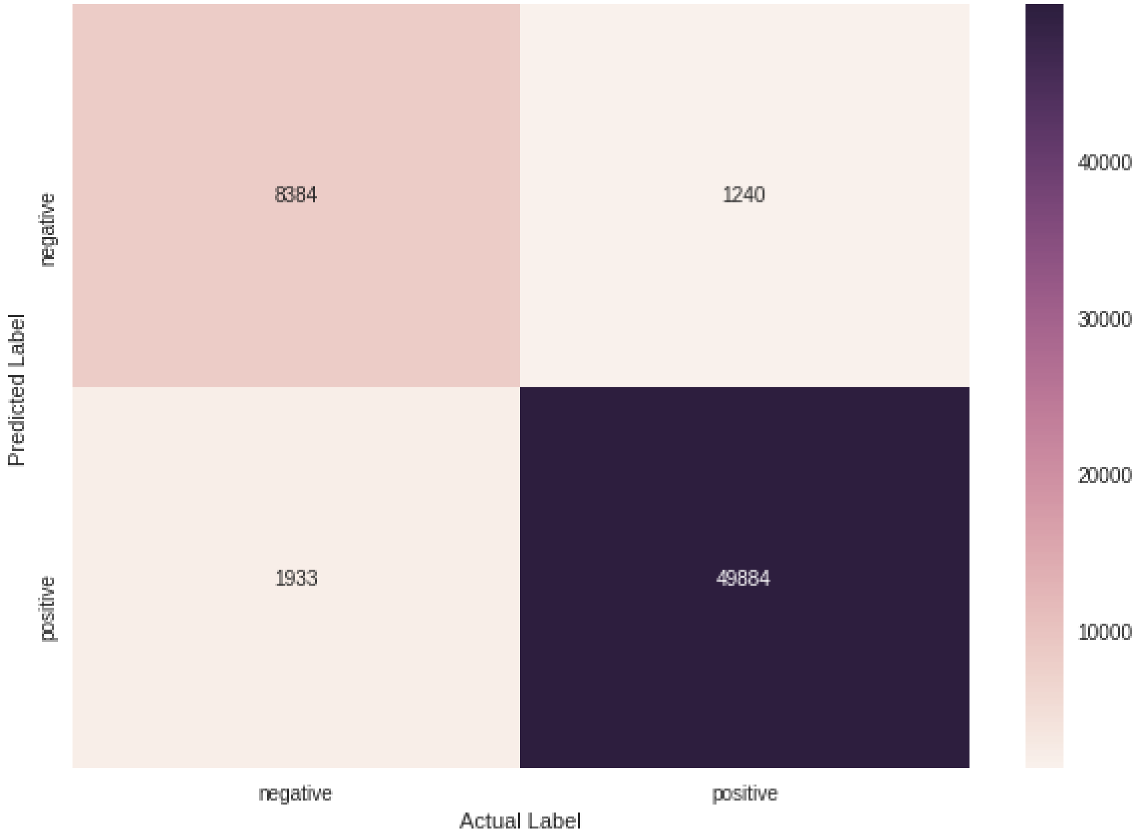
In [105]:

```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, optimal_NB.predict(X_Train_BOW))

df_cm = pd.DataFrame(array, index = [i for i in class_names],columns =
[i for i in class_names])
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True,fmt="d")
plt.xlabel("Actual Label")
plt.ylabel("Predicted Label")
plt.show()
```

```
Train confusion matrix
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

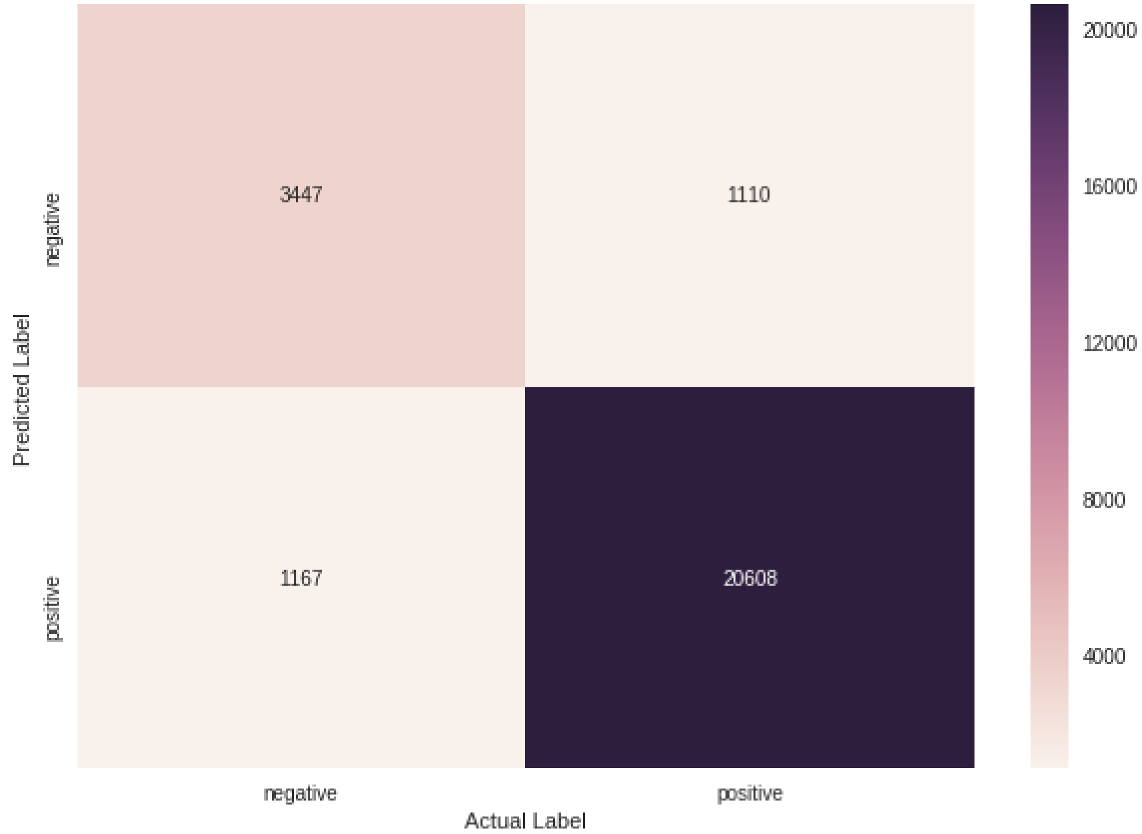


In [106]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, optimal_NB.predict(X_Test_BOW))

df_cm = pd.DataFrame(array, index = [i for i in class_names],columns =
[i for i in class_names])
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True,fmt="d")
plt.xlabel("Actual Label")
plt.ylabel("Predicted Label")
plt.show()
```

```
Test confusion matrix
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



[5.2] Applying Naive Bayes on TFIDF, SET 2

In [107]:
```
# Please write all the code with proper documentation

alpha= [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]

from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV

nb=MultinomialNB()
parameters= {'alpha':alpha}
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
clf=GridSearchCV(nb, parameters, cv=3, scoring='roc_auc')
clf.fit(X_Train_TFIDF, Y_Train)

train_auc=clf.cv_results_['mean_train_score']
train_auc_std=clf.cv_results_['std_train_score']
cv_auc=clf.cv_results_['mean_test_score']
cv_auc_std=clf.cv_results_['std_test_score']

plt.plot(alpha, train_auc, label='Train AUC')

plt.plot(alpha, cv_auc, label='CV AUC')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.xscale('log')
plt.title("ERROR PLOTS")
plt.show()
```

In [108]:
```
print(clf.best_params_)
print(clf.best_estimator_)
print(clf.best_score_)
```

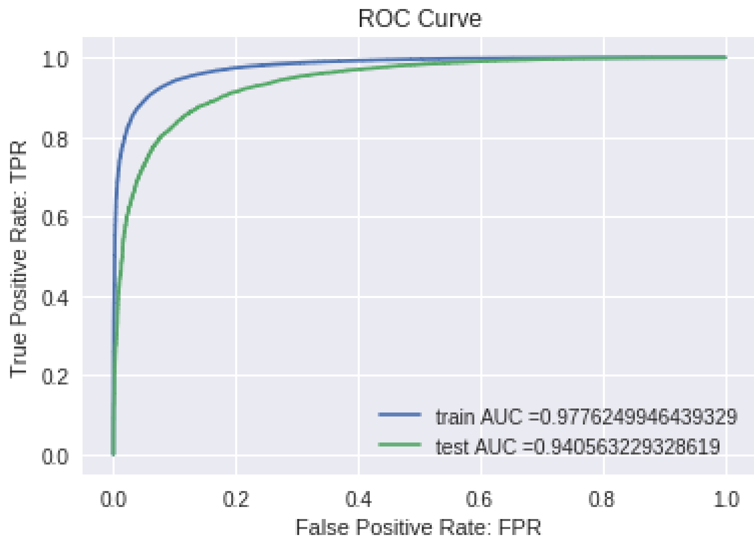
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
{'alpha': 0.1}
MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
0.9251392170134956
```

In [109]:
```
optimal_NB=MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
optimal_NB.fit(X_Train_TFIDF, Y_Train)

train_fpr, train_tpr, thresholds= roc_curve(Y_Train, optimal_NB.predict
_proba(X_Train_TFIDF)[:,1])
test_fpr, test_tpr, thresholds=roc_curve(Y_Test, optimal_NB.predict_pro
ba(X_Test_TFIDF)[:,1])
train_TFIDF_acc =auc(train_fpr, train_tpr)
test_TFIDF_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(train_TFIDF_acc
))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(test_TFIDF_acc))
plt.legend()
plt.xlabel("False Positive Rate: FPR")
plt.ylabel("True Positive Rate: TPR")
plt.title("ROC Curve")
plt.show()
```



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

In [110]:
```
# Please write all the code with proper documentation

class_features=optimal_NB.feature_log_prob_
#  row_0 is for 'negative' class and row_1 is for 'positive' class
negative_features=class_features[0]
positive_features=class_features[1]

feature_names=tf_idf_vect.get_feature_names()

# Sorting 'positive_features' in descending order using argsort()functi
on. sorted_positive_features holds the index of top +ve features

sorted_positive_features=np.argsort(positive_features)[::-1]

print("\n\nTop 10 Important Features and their log probabilities For Po
sitive Class :\n\n")

for i in list(sorted_positive_features[0:10]):
  print("%s\t -->\t%f  "%(feature_names[i] , positive_features[i]))
```

```
Top 10 Important Features and their log probabilities For Positive Clas
s :


not      -->    -4.910125
great    -->    -4.978739
good     -->    -5.131800
tea      -->    -5.269179
coffee   -->    -5.273769
love     -->    -5.340229
like     -->    -5.359322
best     -->    -5.530332
product  -->    -5.544479
taste    -->    -5.564590
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

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

In [111]:
```
# Please write all the code with proper documentation

# Sorting 'positive_features' in descending order using argsort()functi
on. sorted_negative_features holds the index of top -ve features

sorted_negative_features=np.argsort(negative_features)[::-1]

print("\n\nTop 10 Important Features and their log probabilities For Ne
gative Class :\n\n")

for i in list(sorted_negative_features[0:10]):
  print("%s\t -->\t%f  "%(feature_names[i] , negative_features[i]))
```

```
Top 10 Important Features and their log probabilities For Negative Clas
s :


not      -->    -4.333759
like     -->    -5.178132
taste    -->    -5.262192
would    -->    -5.381201
product  -->    -5.384952
coffee   -->    -5.477971
one      -->    -5.675929
flavor   -->    -5.733125
no       -->    -5.774436
good     -->    -5.791238
```

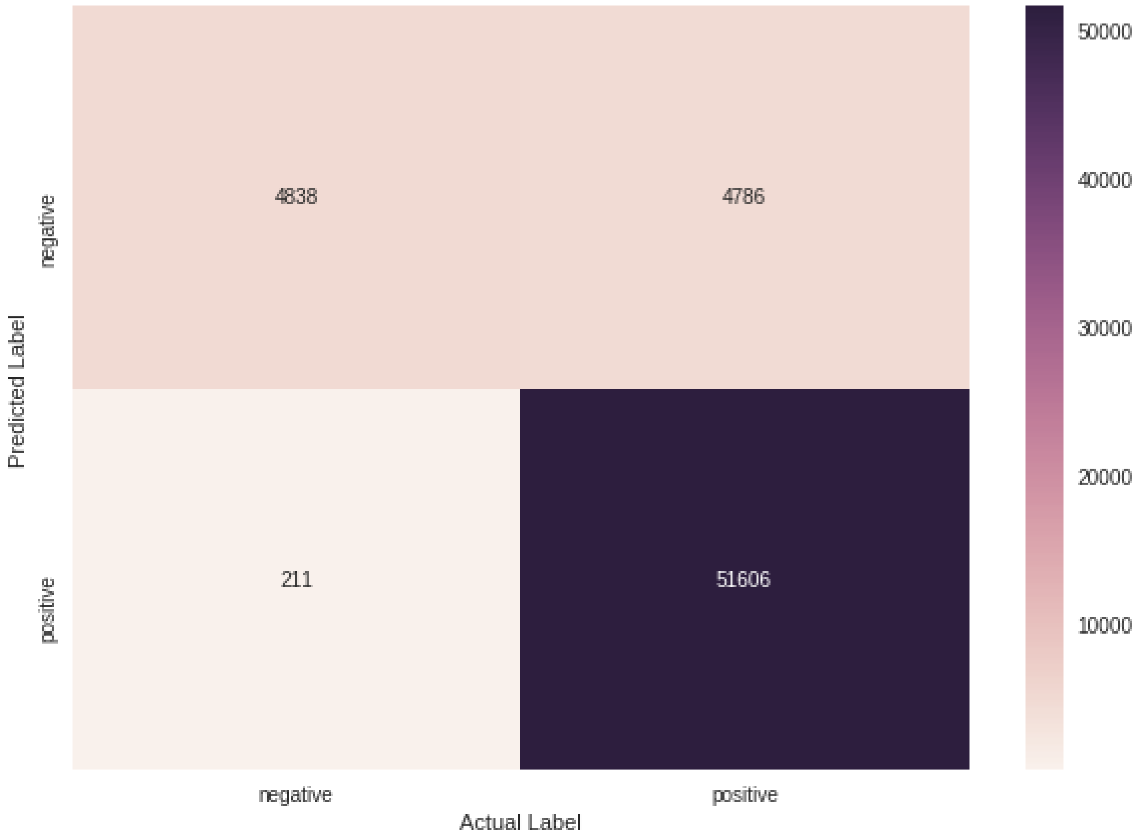
In [112]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, optimal_NB.predict(X_Train_TFIDF))

df_cm = pd.DataFrame(array, index = [i for i in class_names],columns =
[i for i in class_names])
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True,fmt="d")
plt.xlabel("Actual Label")
plt.ylabel("Predicted Label")
plt.show()
```

Train confusion matrix



In [113]:

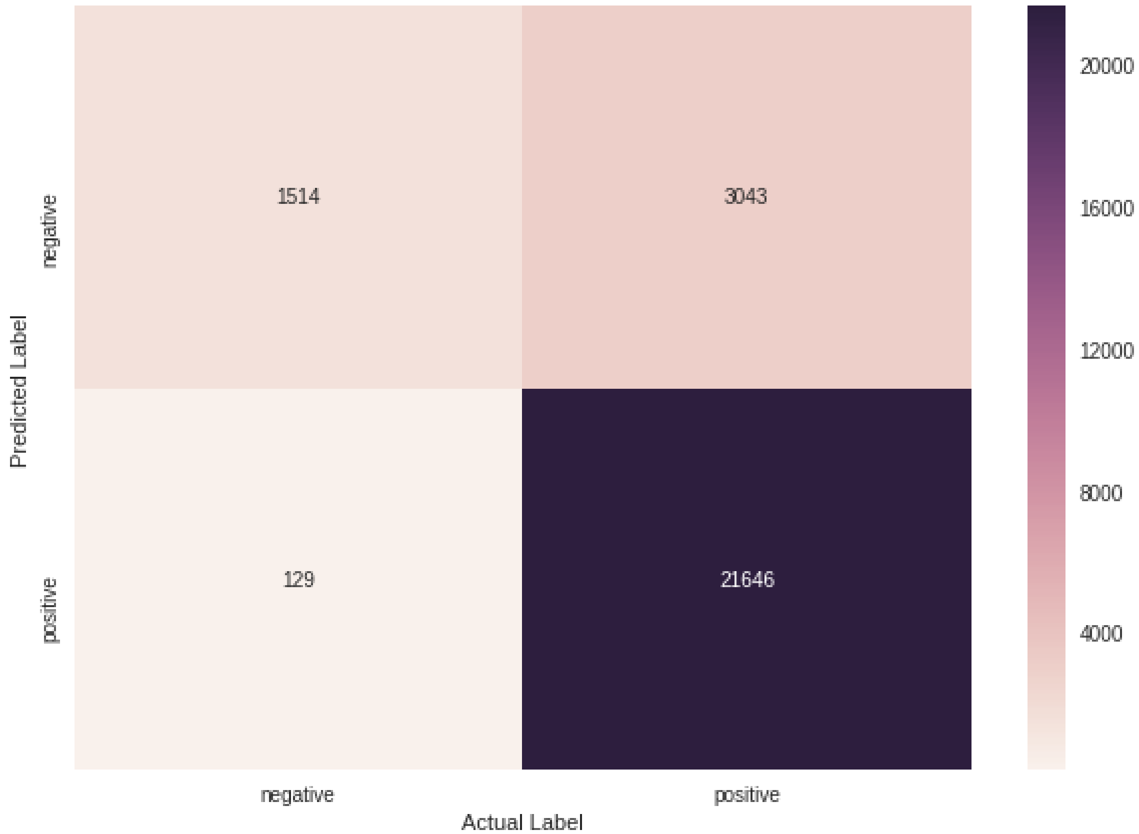
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, optimal_NB.predict(X_Test_TFIDF))

df_cm = pd.DataFrame(array, index = [i for i in class_names],columns =
[i for i in class_names])
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
plt.figure(figsize = (10,7))
sns.heatmap(df_cm, annot=True,fmt="d")
plt.xlabel("Actual Label")
plt.ylabel("Predicted Label")
plt.show()
```

Test confusion matrix



[6] Conclusions

In [114]:
```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
names= [
        "Naive Bayes using BoW",
                    "Naive Bayes using TFIDF",

                ]


optimal_Alpha= [0.1,0.1]

train_acc= [
                        train_bow_acc,
                        train_TFIDF_acc,
                        ]

test_acc = [
                        test_bow_acc,
                        test_TFIDF_acc,
 ]
numbering= [1,2]


# Initializing prettytable

ptable=PrettyTable()

 # Adding columns
ptable.add_column("S.NO.",numbering)
ptable.add_column("MODEL",names)
ptable.add_column("Best Alpha",optimal_Alpha)
ptable.add_column("Training Accuracy",train_acc)
ptable.add_column("Test Accuracy",test_acc)

# Printing the Table
print(ptable)
```

```
+-------+------------------------+-----------+-------------------+--
------------------+
| S.NO. |         MODEL          | Best Alpha | Training Accuracy  |
Test Accuracy    |
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
+-------+-----------------------+-----------+------------------+--
-----------------+
|   1   |  Naive Bayes using BoW  |    0.1     | 0.9786779862041187 |
0.9324591958927195 |
|   2   | Naive Bayes using TFIDF |    0.1     | 0.9776249946439329 |
0.940563229328619  |
+-------+-----------------------+-----------+------------------+--
-----------------+
```

Conclusion :

1. I have taken 100000 data from SQL database.
2. After that I have cleaned the summary and text of the reviews.
3. Merge the Summary and text into new text column.
4. Sort the dataset based on time.
5. Split the dataset into Train and Test based on time.
6. Applied BOW, TFIDF on review text.
7. Applied MultinomialNB on BOW andTFIDF.
8. Inplemented grid Search.
9. Implemented confusion matrix and AOC to see the TP and TN. .

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD