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).

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

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

[1]. Reading Data

[1.1] Loading the data

The dataset is available in two forms

1. .csv file
2. SQLite Database

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

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

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


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

```
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
from google.colab import drive


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: python-dateutil>=2 in /usr/local/lib/pyt
```

```
hon3.6/dist-packages (from pandas->pandasql) (2.5.3)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/
dist-packages (from pandas->pandasql) (2018.9)
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) (3.0.0)
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 [3]:

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

No file chosen

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[3]:
```
{'kaggle.json': b'{"username":"pranabdas457","key":"8d3a5d1700d8cdee57b
f3120ba69c59e"}'}
```

In [0]:
```
#before importing the dataset we want to use this code
# The Kaggle API client expects this file to be in ~/.kaggle,
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

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

In [5]:
```
!ls
```

```
kaggle.json  sample_data
```

In [6]:
```
#import the dataset we want to use for our project
!kaggle datasets download -d snap/amazon-fine-food-reviews --force
```

```
Downloading amazon-fine-food-reviews.zip to /content
 98% 246M/251M [00:02<00:00, 135MB/s]
100% 251M/251M [00:02<00:00, 110MB/s]
```

In [7]:
```
!ls
```

```
amazon-fine-food-reviews.zip  kaggle.json  sample_data
```

In [8]:
```
!unzip amazon-fine-food-reviews.zip
```

```
Archive:  amazon-fine-food-reviews.zip
  inflating: Reviews.csv
  inflating: database.sqlite
  inflating: hashes.txt
```

In [9]:
```
!ls
```

```
amazon-fine-food-reviews.zip  hashes.txt   Reviews.csv
database.sqlite               kaggle.json  sample_data
```

In [10]:
```
# 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 Score
 != 3 LIMIT 70000""", con)
# for tsne assignment you can take 5k data points

#filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Scor
e != 3 LIMIT 5000""", 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
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(5)
```

```
Number of data points in our data (70000, 10)
```

Out[10]:

	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
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulnes
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0

In [0]:
```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [12]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[12]:

	UserId	ProductId	ProfileName	Time	Score	Text	COU
0	#oc-R115TNMSPFT9I7	B007Y59HVM	Breyton	1331510400	2	Overall its just OK when considering the price...	2

	UserId	ProductId	ProfileName	Time	Score	Text	COU
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 [13]:
```
display[display['UserId']=='#oc-R11D9D7SHXIJB9']
```

Out[13]:

	UserId	ProductId	ProfileName	Time	Score	Text	COUNT(*)

	UserId	ProductId	ProfileName	Time	Score	Text	COUNT(*)
1	#oc-R11D9D7SHXIJB9	B005HG9ET0	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3

In [14]:
```
display['COUNT(*)'].sum()
```

Out[14]: 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 [15]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[15]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln

	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
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=Tr
ue, inplace=False, kind='quicksort', na_position='last')
```

In [17]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time"
,"Text"}, keep='first', inplace=False)
final.shape
```

Out[17]:
```
(62864, 10)
```

In [18]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[18]:
```
89.80571428571429
```

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 [19]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[19]:

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

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

In [22]:
```
#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()
```

```
(62862, 10)
```

Out[22]:
```
1    52600
0    10262
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 [23]:
```
#Finding the reviews which contains URLs in it.
def Find_Url_In_Review(reviews = [], *args):
    count = 0;
    store5Text = {}
    for idx, val in enumerate(reviews):
```

```
        text = re.findall('https?://(?:[-\w.]|(?:%[\da-fA-F]{2}))+', va
l)
        if(text):
            store5Text[idx] = text
            count+=1
            if(count == 25):
                break

    return store5Text

lst = final['Text'].values.tolist()
print(Find_Url_In_Review(lst))
```

```
{3: ['http://www.amazon.com'], 158: ['http://www.consumeraffairs.com'],
179: ['http://www.amazon.com', 'http://www.amazon.com', 'http://www.ama
zon.com'], 219: ['http://www.amazon.com'], 251: ['http://www.amazon.co
m'], 270: ['http://www.amazon.com'], 341: ['http://www.amazon.com'], 54
3: ['http://www.amazon.com'], 610: ['http://www.amazon.com'], 612: ['ht
tp://www.amazon.com'], 638: ['http://www.amazon.com', 'http://www.amazo
n.com', 'http://www.amazon.com'], 667: ['http://www.amazon.com'], 724:
['http://www.amazon.com'], 818: ['http://www.amazon.com'], 826: ['htt
p://www.amazon.com'], 842: ['http://www.amazon.com'], 999: ['http://ww
w.amazon.com'], 1026: ['http://www.amazon.com'], 1135: ['http://www.tij
uanaflats.com'], 1249: ['http://www.amazon.com', 'http://www.amazon.co
m'], 1264: ['http://www.amazon.com'], 1281: ['http://www.amazon.com',
'http://www.amazon.com', 'http://www.amazon.com', 'http://www.amazon.co
m', 'http://www.amazon.com', 'http://www.amazon.com', 'http://www.amazo
n.com', 'http://www.amazon.com'], 1435: ['http://www.amazon.com'], 145
8: ['http://www.amazon.com'], 1482: ['http://www.amazon.com']}
```

In [24]:

```
# printing some random reviews
sent_179 = final['Text'].values[179]
print(sent_179)
print("="*50)

sent_158 = final['Text'].values[158]
print(sent_158)
print("="*50)
```

```
sent_1264 = final['Text'].values[1264]
print(sent_1264)
print("="*50)

sent_1281 = final['Text'].values[1281]
print(sent_1281)
print("="*50)
```

```
These wholesome treats are only 1 or 2 calories per biscotti "crouton."
My 50 pound dog has a weight problem, and these are the only treats I c
an use to reward him for obedience multiple times a day.<br /><br />The
biscotti don't crumble in my  pocket on a walk, and my dog even sniffs
my pockets at home to ask for another one. Even though the treats are t
iny, he doesn't swallow them whole. He chews them--maybe to savor them.
I also put these in his <a href="http://www.amazon.com/gp/product/B0038
WP1YC">Ethical Pet Seek-A-Treat Shuffle Bone Dog Puzzle</a> as a rewar
d, and he solves his puzzle with enthusiasm. Even though my dog weighs
50 pounds, one 8 oz bag lasts about a month.<br /><br />My dog has no d
iagnosed allergies, but his gasto-esophgial system was sensitive and ir
regular starting from the time we rescued him. For his weight-loss, I s
witched his food to <a href="http://www.amazon.com/gp/product/B003P9XFV
4">Blue Buffalo Longevity Dry Food for Adult Dogs, 24-Pound Bag</a>, wh
ich has no soy, wheat, corn, by-products, or preservatives. His gastro-
esophogeal system happened to improve a lot and became regular. The hyp
o-allergenic treats did not cost much more than the regular ones at the
time I first bought them, so I figured I would be safe and buy them. My
dog hasn't had any problems on these treats, but the change in food mig
ht also explain it.  I'm sticking with the hypo-allergenic even though
it's not eligible for free shipping.<br /><br />Having said that my dog
likes these treats a lot, these are not his very favorite.  I give him
a medium-sized biscuit <a href="http://www.amazon.com/gp/product/B00027
D3QO">Nutri-Vet Breath & Tartar Spearmint & Parsley Flavored Biscuits,
19.5 Ounce Bag</a>every few days, and he goes absolutely crazy for it.
Alas, the biscuits have too many calories for me to treat him more ofte
n. I'd say my dog would rate his biscuits 10 barks for taste, whereas h
e would give his biscotti a solid 7 barks.<br /><br />Because the bisco
tti treats are tasty, made with wholesome ingredients, and very low in
calories, I rate them 5 stars.  I consider liver biscotti a good invest
ment in my dog's health. Every dog should have liver biscotti as his or
her go-to treats.
```

```
==================================================
I used to be a fan of Canidae, but after $2k in recent vet bills and a
very sick dog, I found out there is a HUGE amount of Consumer Affairs c
omplaints about this company.<br /><br />After blood work, ultrasound a
nd many other tests trying to figure out why my dog wasn't eating, was
throwing up on the rare occasion he did, had pockets of gas the size of
base balls (he is a small dog) and had all the symptoms of kidney and l
iver failure, my vet determined it might be the food. I found that hard
to believe, but we couldn't find anything else wrong. If you read the c
omplaints you will see many people have lost their dogs. They started u
sing one of the very companies for production whose imported food kille
d so many animals a year or so ago.<br /><br />As soon as I quit feedin
g him this poison, he started getting better. I was one of the lucky on
es since my pet did not die, but as I write this, he still has not gain
ed all his weight back.<br /><br />The company is acting as if nothing
is wrong. Probably on the advice of their attorneys.<br /><br />You do
n't need to take my word for it, please read up... http://www.consumera
ffairs.com/pets/canidae.html
==================================================
Green chili sauces tend to be milder than their red companions, and thi
s "Castillo Salsa Habanera - Green" is no exception.  It has a nice, ti
ny little kick to it, but it is mild enough that I could just pour it s
traight into my mouth.<br /><br />So, unless you don't have much tolera
nce for heat, you are buying this for the flavor, not the spice. And th
is does have a nice flavor.  It is a good blend of habanera, garlic, an
d some other spices, without being too vinegary or overpowering.  We ha
d some fajitas for breakfast this morning, and this Castillo Salsa was
poured quite liberally over them.  Delicious.<br /><br />It isn't quite
as good as my favorite green salsa,<a href="http://www.amazon.com/gp/pr
oduct/B0000DIWWE">Blair's Jalapeno Death Sauce with Tequila</a> , but a
bottle of this Castillo sauce can usually be found in my kitchen.
==================================================
QUALITY<br />In my opinion, having used a dozen or so brands of olive o
il, Colavita is simply the best reasonably priced olive oil which is wi
dely available.  I've been using Colavita for several years now, and ha
ve never been dissappointed.  Colavita is excellent quality, dependabl
e, and reasonably priced---the perfect combination for a daily general
purpose olive oil.<br /><br />FLAVOR<br />Although not an "olive oil go
urmand", I'd describe Colavita as "rich and smooth".  Connoisseurs prob
```

ably have other silly terms for the flavor, such as "nutty" or "fruity" (but a "civilian" would never describe the flavor as nutty or fruity). It is not spicy like Whole Foods 360, and NOT strongly "olive-oily" (i.e., "herby"). Colavita is an evoo ("extra virgin olive oil"---to us non-olive oil snobs), which means "first pressing". Most olive-oil nut s consider the words "extra virgin" to be sacred. Practically speaking though, the mere label "extra virgin" is not a guarantee of quality---m ost grocery store brands (evoo or not) are harsh and bitter, but some a re good cooking oils.

PREMIUM OILS
Botique premium oils are are as variable year to year as wines, comparably expensive, and ha ve to be bought the way you buy fine wine (that is, ideally at tasting s, where you confirm the quality and flavor before purchase). Some are genuinely (as in NOT-gourmet-speak) buttery and nutty---great for dippi ng. Some taste like the fragrance of flowers---great for salads. Neve r cook premium olive oil---use it for dipping or drizzling, or on delic ate salads. Colavita is a very good general purpose oil, perhaps the b est common grocery store olive oil, but is not quite a premium oil.

Although heresy (comparable to suggesting fruit wines to a wine connoisseur) there are inexpensive alternatives to premium botique oliv e oils, including pumpkinseed, almond, sesame, and walnut oil. I parti cularly love pumpkinseed oil as a dipping oil (by itself, no vinegar, a nd with a little salt). The following are examples---I can't vouch for the specific brands as they are different from what I use. Styrian Pumpkinseed Oil 8.45 oz, Pure Almo nd Oil 300ml, Kadoya - Pure Sesame Oil 5.5 Oz., 100% Natural Walnut Oil, Cold Pressed 16 fl oz (474 ml) Liquid

COOKING OIL
Non-extra virgin olive oils (second or later pressings) are usually darker and stronger flavor ed---which adds depth and complexity to soups, sauses, stews, chili, ca sseroles, bean dishes etc. I add 1/4 to 1/2 cup of olive oil to such di shes. Unless you particularly want the dish to taste "olive-oily" (I n ever do), cook the oil WITH the dish. However, olive oil can "go bad"- --so unless you use alot, it doesn't sense to use a different cooking o il than you use on salads, for dipping etc. Colavita is a very good ge neral purpose oil, and therefore a good cooking oil.

"Cookin g" does NOT mean "frying" with ANY olive oil. Because of the low smoke point (i.e., olive oil burns at a low temperature) olive oil is NOT sui

table for frying. That said, you can carefully fry an egg in olive oil at a low temperature (and honestly, good fried eggs swimming in good ol ive oil are wonderful). But you can NOT fry meat nor deep-fry anythin g. The best frying oil is peanut oil. Canola is a good frying oil and is particularly healthy. Like olive oil there is variation in the bran ds. Peanut oil should be slightly nutty, but not peanutty. Canola oil should be flavorless.

DIPPING OIL
Traditionally (and in the best restaurants), Italian bread, oil and balsamic vinegar is serve d as an appetizer. The diner usually mixes his own oil and vinegar at the table. Albeit probably heresy, you can subsitute other premium typ es of vinegar (such as wine vinegar, rasberry vinegar, or even apple vi negar) or fresh lemon juice. I usually add a few drops soy sause or Br aggs for saltiness and richness to the mixture. Bragg Liquid Aminos, Natural Soy Sauce A lternative, 32-Ounce Bottle , (Pack of 3) Colavita is a very good "everyday" dipping oil.

SALAD OIL
The requirements for an "everyday" salad oil are a little less stringent than that for a dip ping oil, because the heavy flavor of the vinegar, and often pamessan, bacon bits, etc. usually dominate. However, I rarely use oil and vineg ar the traditional way. For everyday salads (or as a dip for raw vegie s) I prefer olive-oil based mayo with a little freshly ground pepper, a nd some citrus zest. For special salads, I use a seed or nut oil or a fruity premium olive oil (and just a hint of lemon and/or orange juice and zest). Colavita is a very good "everyday" salad oil. If you are n ot familiar with fresh citrus zest (wonderful stuff), the tool to use i s Microplane 4002 0 Classic Zester/Grater

DRIZZING OIL
A drizzling oi l is an oil you pour over a dish immediately before serving, or which y ou serve in a carafe on the table. If you REALLY like the flavor of "o rdinary good" olive oil (I don't)---then you will find Colavita to be a good drizzling oil. My hispanic wife frequently drizzles Colavita on t he dishes I've cooked (with Colavita), because she likes the olive-oil flavor. By the way, the same principle applies to the use of butter, p epper, garlic, cheese, even soy sause---if you want the flavor to domin ate, add it just before serving; if you want the flavor to be subtle, c ook the seasoning with the dish. For most dishes I prefer a rich blend of subtle flavors.

HEALTHY COOKING
I go to a great deal of effort to minimize animal fat in my diet, by very strict trimming, t horough cooking, and by chilling broth to remove the solidified fat. H

owever, the body needs fat and craves fat. The ideal solution is to re
place animal fat with healthy vegetable oils. Olive oil is ideal for t
he purpose. Canola is better, but is essentially flavorless.

My Southern grandmother could not cook any vegetables without a thick
slice of fatback (fatty bacon)---and indeed her vegetables were very ta
sty. Speaking of which, what is the point of healthy vegetables if no-
one will eat them? The solution is to add several tablespoon fulls of
olive oil, and just a teaspoon of chipped ham <a href="http://www.amazo
n.com/gp/product/B000NERTSE">Hormel Premium Real Crumbled Bacon to
the vegies. For the best flavor, cook the vegies with the olive oil (d
o NOT add oil to the cooked vegetables---unless you really like the fla
vor of the oil). Season as necessary with pepper and fresh lemon juic
e. Truly delicious vegetables are possible---I eat them daily.
<b
r />ECONOMY
It has been about a year since I've purchased Colavita
because I can buy a better but obscure brand (Bella Famiglia) locally f
or $12 per 17oz bottle. The 34 ounce tins of Colavita <a href="http://
www.amazon.com/gp/product/B001ELLB16">Colavita Extra Virgin Olive Oil,
34-Ounce Tins (Pack of 2) are about half the price of the bottle pe
r oz at this time, but is still expensive for the quality, particularly
with the shipping charge. The tins are small enough to use directly. B
ut I prefer bottles, because controlling the flow of the oil is easier
(since you can see the oil approaching the spout), and it is easier to
gauge the amount of oil I'm using by eye (how far the level drops in th
e glass bottle). So, I refill bottles from the tins. I've been told th
at olive oil stays fresher in tins (measured in months and years) That
becomes relevant when you don't know how many months (or even years) a
bottle may have sat on a grocery shelf.

GENERAL RECCOMENDATI
ON
The price of many items, including Colavita vary wildly on Amaz
on, sometimes from day-to-day. Sometimes it is available directly from
Amazon with free shipping, sometimes not---which makes a big difference
in the total coat. After major grocery shopping expeditions, sit down
at your computer with your reciept, and check if you can buy any of the
non-perishables through Amazon. When you find items (even if more expen
sive than you just paid), put the item on your Amazon wish list, and ad
d a note to the wish list of the price you just paid. Check your wish l
ist frequently, and when you see an item you need at a bargain price, b
uy it.
==========

In [25]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/40
84039
sent_179 = re.sub(r"http\S+", "", sent_179)
sent_158 = re.sub(r"http\S+", "", sent_158)
sent_1264 = re.sub(r"http\S+", "", sent_1264)
sent_1281 = re.sub(r"http\S+", "", sent_1281)

print(sent_179)
print("="*50)
print(sent_158)
print("="*50)
print(sent_1264)
print("="*50)
print(sent_1281)
print("="*50)
```

```
These wholesome treats are only 1 or 2 calories per biscotti "crouton."
My 50 pound dog has a weight problem, and these are the only treats I c
an use to reward him for obedience multiple times a day.<br /><br />The
biscotti don't crumble in my  pocket on a walk, and my dog even sniffs
my pockets at home to ask for another one. Even though the treats are t
iny, he doesn't swallow them whole. He chews them--maybe to savor them.
I also put these in his <a href=" Pet Seek-A-Treat Shuffle Bone Dog Puz
zle</a> as a reward, and he solves his puzzle with enthusiasm. Even tho
ugh my dog weighs 50 pounds, one 8 oz bag lasts about a month.<br /><br
/>My dog has no diagnosed allergies, but his gasto-esophgial system was
sensitive and irregular starting from the time we rescued him. For his
weight-loss, I switched his food to <a href=" Buffalo Longevity Dry Foo
d for Adult Dogs, 24-Pound Bag</a>, which has no soy, wheat, corn, by-p
roducts, or preservatives. His gastro-esophogeal system happened to imp
rove a lot and became regular. The hypo-allergenic treats did not cost
much more than the regular ones at the time I first bought them, so I f
igured I would be safe and buy them. My dog hasn't had any problems on
these treats, but the change in food might also explain it.  I'm sticki
ng with the hypo-allergenic even though it's not eligible for free ship
ping.<br /><br />Having said that my dog likes these treats a lot, thes
e are not his very favorite.  I give him a medium-sized biscuit <a href
=" Breath & Tartar Spearmint & Parsley Flavored Biscuits, 19.5 Ounce Ba
g</a>every few days, and he goes absolutely crazy for it. Alas, the bis
cuits have too many calories for me to treat him more often. I'd say my
```

dog would rate his biscuits 10 barks for taste, whereas he would give h
is biscotti a solid 7 barks.

Because the biscotti treats are
tasty, made with wholesome ingredients, and very low in calories, I rat
e them 5 stars. I consider liver biscotti a good investment in my do
g's health. Every dog should have liver biscotti as his or her go-to tr
eats.
==========
I used to be a fan of Canidae, but after $2k in recent vet bills and a
very sick dog, I found out there is a HUGE amount of Consumer Affairs c
omplaints about this company.

After blood work, ultrasound a
nd many other tests trying to figure out why my dog wasn't eating, was
throwing up on the rare occasion he did, had pockets of gas the size of
base balls (he is a small dog) and had all the symptoms of kidney and l
iver failure, my vet determined it might be the food. I found that hard
to believe, but we couldn't find anything else wrong. If you read the c
omplaints you will see many people have lost their dogs. They started u
sing one of the very companies for production whose imported food kille
d so many animals a year or so ago.

As soon as I quit feedin
g him this poison, he started getting better. I was one of the lucky on
es since my pet did not die, but as I write this, he still has not gain
ed all his weight back.

The company is acting as if nothing
is wrong. Probably on the advice of their attorneys.

You do
n't need to take my word for it, please read up...
==========
Green chili sauces tend to be milder than their red companions, and thi
s "Castillo Salsa Habanera - Green" is no exception. It has a nice, ti
ny little kick to it, but it is mild enough that I could just pour it s
traight into my mouth.

So, unless you don't have much tolera
nce for heat, you are buying this for the flavor, not the spice. And th
is does have a nice flavor. It is a good blend of habanera, garlic, an
d some other spices, without being too vinegary or overpowering. We ha
d some fajitas for breakfast this morning, and this Castillo Salsa was
poured quite liberally over them. Delicious.

It isn't quite
as good as my favorite green salsa,<a href=" Jalapeno Death Sauce with
Tequila , but a bottle of this Castillo sauce can usually be found
in my kitchen.
==========
QUALITY
In my opinion, having used a dozen or so brands of olive o
il, Colavita is simply the best reasonably priced olive oil which is wi
dely available. I've been using Colavita for several years now, and ha

ve never been dissappointed. Colavita is excellent quality, dependabl
e, and reasonably priced---the perfect combination for a daily general
purpose olive oil.

FLAVOR
Although not an "olive oil go
urmand", I'd describe Colavita as "rich and smooth". Connoisseurs prob
ably have other silly terms for the flavor, such as "nutty" or "fruity"
(but a "civilian" would never describe the flavor as nutty or fruity).
It is not spicy like Whole Foods 360, and NOT strongly "olive-oily"
(i.e., "herby"). Colavita is an evoo ("extra virgin olive oil"---to us
non-olive oil snobs), which means "first pressing". Most olive-oil nut
s consider the words "extra virgin" to be sacred. Practically speaking
though, the mere label "extra virgin" is not a guarantee of quality---m
ost grocery store brands (evoo or not) are harsh and bitter, but some a
re good cooking oils.

PREMIUM OILS
Botique premium oils
are are as variable year to year as wines, comparably expensive, and ha
ve to be bought the way you buy fine wine (that is, ideally at tasting
s, where you confirm the quality and flavor before purchase). Some are
genuinely (as in NOT-gourmet-speak) buttery and nutty---great for dippi
ng. Some taste like the fragrance of flowers---great for salads. Neve
r cook premium olive oil---use it for dipping or drizzling, or on delic
ate salads. Colavita is a very good general purpose oil, perhaps the b
est common grocery store olive oil, but is not quite a premium oil.

Although heresy (comparable to suggesting fruit wines to a wine
connoisseur) there are inexpensive alternatives to premium botique oliv
e oils, including pumpkinseed, almond, sesame, and walnut oil. I parti
cularly love pumpkinseed oil as a dipping oil (by itself, no vinegar, a
nd with a little salt). The following are examples---I can't vouch for
the specific brands as they are different from what I use. <a href=" P
umpkinseed Oil 8.45 oz, <a href=" Almond Oil 300ml, <a href=" -
Pure Sesame Oil 5.5 Oz., <a href=" Natural Walnut Oil, Cold Pressed
16 fl oz (474 ml) Liquid

COOKING OIL
Non-extra virg
in olive oils (second or later pressings) are usually darker and strong
er flavored---which adds depth and complexity to soups, sauses, stews,
chili, casseroles, bean dishes etc. I add 1/4 to 1/2 cup of olive oil t
o such dishes. Unless you particularly want the dish to taste "olive-o
ily" (I never do), cook the oil WITH the dish. However, olive oil can
"go bad"---so unless you use alot, it doesn't sense to use a different
cooking oil than you use on salads, for dipping etc. Colavita is a ver
y good general purpose oil, and therefore a good cooking oil.

"Cooking" does NOT mean "frying" with ANY olive oil. Because of the

low smoke point (i.e., olive oil burns at a low temperature) olive oil is NOT suitable for frying. That said, you can carefully fry an egg in olive oil at a low temperature (and honestly, good fried eggs swimming in good olive oil are wonderful). But you can NOT fry meat nor deep-fry anything. The best frying oil is peanut oil. Canola is a good frying oil and is particularly healthy. Like olive oil there is variation in the brands. Peanut oil should be slightly nutty, but not peanutty. Canola oil should be flavorless.

DIPPING OIL
Traditionally (and in the best restaurants), Italian bread, oil and balsamic vinegar is served as an appetizer. The diner usually mixes his own oil and vinegar at the table. Albeit probably heresy, you can subsitute other premium types of vinegar (such as wine vinegar, rasberry vinegar, or even apple vinegar) or fresh lemon juice. I usually add a few drops soy sause or Braggs for saltiness and richness to the mixture. <a href=" Liquid Aminos, Natural Soy Sauce Alternative, 32-Ounce Bottle , (Pack of 3) Colavita is a very good "everyday" dipping oil.

SALAD OIL
The requirements for an "everyday" salad oil are a little less stringent than that for a dipping oil, because the heavy flavor of the vinegar, and often pamessan, bacon bits, etc. usually dominate. However, I rarely use oil and vinegar the traditional way. For everyday salads (or as a dip for raw vegies) I prefer olive-oil based mayo with a little freshly ground pepper, and some citrus zest. For special salads, I use a seed or nut oil or a fruity premium olive oil (and just a hint of lemon and/or orange juice and zest). Colavita is a very good "everyday" salad oil. If you are not familiar with fresh citrus zest (wonderful stuff), the tool to use is <a href=" 40020 Classic Zester/Grater

DRIZZING OIL
A drizzling oil is an oil you pour over a dish immediately before serving, or which you serve in a carafe on the table. If you REALLY like the flavor of "ordinary good" olive oil (I don't)---then you will find Colavita to be a good drizzling oil. My hispanic wife frequently drizzles Colavita on the dishes I've cooked (with Colavita), because she likes the olive-oil flavor. By the way, the same principle applies to the use of butter, pepper, garlic, cheese, even soy sause---if you want the flavor to dominate, add it just before serving; if you want the flavor to be subtle, cook the seasoning with the dish. For most dishes I prefer a rich blend of subtle flavors.

HEALTHY COOKING
I go to a great deal of effort to minimize animal fat in my diet, by very strict trimming, thorough cooking, and by chilling broth to remove the solidified fat. However, the body needs fa

```
t and craves fat.  The ideal solution is to replace animal fat with hea
lthy vegetable oils.  Olive oil is ideal for the purpose.  Canola is be
tter, but is essentially flavorless.<br /><br />My Southern grandmother
could not cook any vegetables without a thick slice of fatback (fatty b
acon)---and indeed her vegetables were very tasty.  Speaking of which,
what is the point of healthy vegetables if no-one will eat them?  The s
olution is to add several tablespoon fulls of olive oil, and just a tea
spoon of chipped ham <a href=" Premium Real Crumbled Bacon</a> to the v
egies.  For the best flavor, cook the vegies with the olive oil (do NOT
add oil to the cooked vegetables---unless you really like the flavor of
the oil).  Season as necessary with pepper and fresh lemon juice.  Trul
y delicious vegetables are possible---I eat them daily.<br /><br />ECON
OMY<br />It has been about a year since I've purchased Colavita because
I can buy a better but obscure brand (Bella Famiglia) locally for $12 p
er 17oz bottle.  The 34 ounce tins of Colavita <a href=" Extra Virgin O
live Oil, 34-Ounce Tins (Pack of 2)</a> are about half the price of the
bottle per oz at this time, but is still expensive for the quality, par
ticularly with the shipping charge.  The tins are small enough to use d
irectly. But I prefer bottles, because controlling the flow of the oil
is easier (since you can see the oil approaching the spout), and it is
easier to gauge the amount of oil I'm using by eye (how far the level d
rops in the glass bottle). So, I refill bottles from the tins.  I've be
en told that olive oil stays fresher in tins (measured in months and ye
ars) That becomes relevant when you don't know how many months (or even
years) a bottle may have sat on a grocery shelf.<br /><br />GENERAL REC
COMENDATION<br />The price of many items, including Colavita vary wildl
y on Amazon, sometimes from day-to-day.  Sometimes it is available dire
ctly from Amazon with free shipping, sometimes not---which makes a big
difference in the total coat.  After major grocery shopping expedition
s, sit down at your computer with your reciept, and check if you can bu
y any of the non-perishables through Amazon. When you find items (even
if more expensive than you just paid), put the item on your Amazon wish
list, and add a note to the wish list of the price you just paid. Check
your wish list frequently, and when you see an item you need at a barga
in price, buy it.
==================================================
```

In [26]:
```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how
```

```
-to-remove-all-tags-from-an-element
from bs4 import BeautifulSoup



soup = BeautifulSoup(sent_179, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_158, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1264, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1281, 'lxml')
text = soup.get_text()
print(text)
```

```
These wholesome treats are only 1 or 2 calories per biscotti "crouton."
My 50 pound dog has a weight problem, and these are the only treats I c
an use to reward him for obedience multiple times a day.The biscotti do
n't crumble in my  pocket on a walk, and my dog even sniffs my pockets
at home to ask for another one. Even though the treats are tiny, he doe
sn't swallow them whole. He chews them--maybe to savor them. I also put
these in his , which has no soy, wheat, corn, by-products, or preservat
ives. His gastro-esophogeal system happened to improve a lot and became
regular. The hypo-allergenic treats did not cost much more than the reg
ular ones at the time I first bought them, so I figured I would be safe
and buy them. My dog hasn't had any problems on these treats, but the c
hange in food might also explain it.  I'm sticking with the hypo-allerg
enic even though it's not eligible for free shipping.Having said that m
y dog likes these treats a lot, these are not his very favorite.  I giv
e him a medium-sized biscuit
==================================================
```

I used to be a fan of Canidae, but after $2k in recent vet bills and a
very sick dog, I found out there is a HUGE amount of Consumer Affairs c
omplaints about this company.After blood work, ultrasound and many othe
r tests trying to figure out why my dog wasn't eating, was throwing up
on the rare occasion he did, had pockets of gas the size of base balls
(he is a small dog) and had all the symptoms of kidney and liver failur
e, my vet determined it might be the food. I found that hard to believ
e, but we couldn't find anything else wrong. If you read the complaints
you will see many people have lost their dogs. They started using one o
f the very companies for production whose imported food killed so many
animals a year or so ago.As soon as I quit feeding him this poison, he
started getting better. I was one of the lucky ones since my pet did no
t die, but as I write this, he still has not gained all his weight bac
k.The company is acting as if nothing is wrong. Probably on the advice
of their attorneys.You don't need to take my word for it, please read u
p...
==========
Green chili sauces tend to be milder than their red companions, and thi
s "Castillo Salsa Habanera - Green" is no exception. It has a nice, ti
ny little kick to it, but it is mild enough that I could just pour it s
traight into my mouth.So, unless you don't have much tolerance for hea
t, you are buying this for the flavor, not the spice. And this does hav
e a nice flavor. It is a good blend of habanera, garlic, and some othe
r spices, without being too vinegary or overpowering. We had some faji
tas for breakfast this morning, and this Castillo Salsa was poured quit
e liberally over them. Delicious.It isn't quite as good as my favorite
green salsa,
==========
QUALITYIn my opinion, having used a dozen or so brands of olive oil, Co
lavita is simply the best reasonably priced olive oil which is widely a
vailable. I've been using Colavita for several years now, and have nev
er been dissappointed. Colavita is excellent quality, dependable, and
reasonably priced---the perfect combination for a daily general purpose
olive oil.FLAVORAlthough not an "olive oil gourmand", I'd describe Cola
vita as "rich and smooth". Connoisseurs probably have other silly term
s for the flavor, such as "nutty" or "fruity" (but a "civilian" would n
ever describe the flavor as nutty or fruity). It is not spicy like Who
le Foods 360, and NOT strongly "olive-oily" (i.e., "herby"). Colavita
is an evoo ("extra virgin olive oil"---to us non-olive oil snobs), whic

h means "first pressing". Most olive-oil nuts consider the words "extr
a virgin" to be sacred. Practically speaking though, the mere label "e
xtra virgin" is not a guarantee of quality---most grocery store brands
(evoo or not) are harsh and bitter, but some are good cooking oils.PREM
IUM OILSBotique premium oils are are as variable year to year as wines,
comparably expensive, and have to be bought the way you buy fine wine
(that is, ideally at tastings, where you confirm the quality and flavor
before purchase). Some are genuinely (as in NOT-gourmet-speak) buttery
and nutty---great for dipping. Some taste like the fragrance of flower
s---great for salads. Never cook premium olive oil---use it for dippin
g or drizzling, or on delicate salads. Colavita is a very good general
purpose oil, perhaps the best common grocery store olive oil, but is no
t quite a premium oil.Although heresy (comparable to suggesting fruit w
ines to a wine connoisseur) there are inexpensive alternatives to premi
um botique olive oils, including pumpkinseed, almond, sesame, and walnu
t oil. I particularly love pumpkinseed oil as a dipping oil (by itsel
f, no vinegar, and with a little salt). The following are examples---I
can't vouch for the specific brands as they are different from what I u
se. , COOKING OILNon-extra virgin olive oils (second or later pressing
s) are usually darker and stronger flavored---which adds depth and comp
lexity to soups, sauses, stews, chili, casseroles, bean dishes etc. I a
dd 1/4 to 1/2 cup of olive oil to such dishes. Unless you particularly
want the dish to taste "olive-oily" (I never do), cook the oil WITH the
dish. However, olive oil can "go bad"---so unless you use alot, it doe
sn't sense to use a different cooking oil than you use on salads, for d
ipping etc. Colavita is a very good general purpose oil, and therefore
a good cooking oil."Cooking" does NOT mean "frying" with ANY olive oil.
  Because of the low smoke point (i.e., olive oil burns at a low temper
ature) olive oil is NOT suitable for frying. That said, you can carefu
lly fry an egg in olive oil at a low temperature (and honestly, good fr
ied eggs swimming in good olive oil are wonderful). But you can NOT fr
y meat nor deep-fry anything. The best frying oil is peanut oil. Cano
la is a good frying oil and is particularly healthy. Like olive oil th
ere is variation in the brands. Peanut oil should be slightly nutty, b
ut not peanutty. Canola oil should be flavorless.DIPPING OILTraditiona
lly (and in the best restaurants), Italian bread, oil and balsamic vine
gar is served as an appetizer. The diner usually mixes his own oil and
vinegar at the table. Albeit probably heresy, you can subsitute other
premium types of vinegar (such as wine vinegar, rasberry vinegar, or ev

en apple vinegar) or fresh lemon juice. I usually add a few drops soy
sause or Braggs for saltiness and richness to the mixture. SALAD OILThe
requirements for an "everyday" salad oil are a little less stringent th
an that for a dipping oil, because the heavy flavor of the vinegar, and
often pamessan, bacon bits, etc. usually dominate. However, I rarely u
se oil and vinegar the traditional way. For everyday salads (or as a d
ip for raw vegies) I prefer olive-oil based mayo with a little freshly
ground pepper, and some citrus zest. For special salads, I use a seed
or nut oil or a fruity premium olive oil (and just a hint of lemon and/
or orange juice and zest). Colavita is a very good "everyday" salad oi
l. If you are not familiar with fresh citrus zest (wonderful stuff), t
he tool to use is HEALTHY COOKINGI go to a great deal of effort to mini
mize animal fat in my diet, by very strict trimming, thorough cooking,
and by chilling broth to remove the solidified fat. However, the body
needs fat and craves fat. The ideal solution is to replace animal fat
with healthy vegetable oils. Olive oil is ideal for the purpose. Cano
la is better, but is essentially flavorless.My Southern grandmother cou
ld not cook any vegetables without a thick slice of fatback (fatty baco
n)---and indeed her vegetables were very tasty. Speaking of which, wha
t is the point of healthy vegetables if no-one will eat them? The solu
tion is to add several tablespoon fulls of olive oil, and just a teaspo
on of chipped ham are about half the price of the bottle per oz at thi
s time, but is still expensive for the quality, particularly with the s
hipping charge. The tins are small enough to use directly. But I prefe
r bottles, because controlling the flow of the oil is easier (since you
can see the oil approaching the spout), and it is easier to gauge the a
mount of oil I'm using by eye (how far the level drops in the glass bot
tle). So, I refill bottles from the tins. I've been told that olive oi
l stays fresher in tins (measured in months and years) That becomes rel
evant when you don't know how many months (or even years) a bottle may
have sat on a grocery shelf.GENERAL RECCOMENDATIONThe price of many ite
ms, including Colavita vary wildly on Amazon, sometimes from day-to-da
y. Sometimes it is available directly from Amazon with free shipping,
sometimes not---which makes a big difference in the total coat. After
major grocery shopping expeditions, sit down at your computer with your
reciept, and check if you can buy any of the non-perishables through Am
azon. When you find items (even if more expensive than you just paid),
put the item on your Amazon wish list, and add a note to the wish list

```
of the price you just paid. Check your wish list frequently, and when y
ou see an item you need at a bargain price, buy it.
```

In [0]:

```python
# 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 [28]:

```python
sent_179 = decontracted(sent_179)
sent_158 = decontracted(sent_158)
sent_1264 = decontracted(sent_1264)
sent_1281 = decontracted(sent_1281)
print(sent_179)
print("="*50)
print(sent_158)
print("="*50)
print(sent_1264)
print("="*50)
print(sent_1281)
print("="*50)
```

```
These wholesome treats are only 1 or 2 calories per biscotti "crouton."
My 50 pound dog has a weight problem, and these are the only treats I c
an use to reward him for obedience multiple times a day.<br /><br />The
```

biscotti do not crumble in my pocket on a walk, and my dog even sniffs
my pockets at home to ask for another one. Even though the treats are t
iny, he does not swallow them whole. He chews them--maybe to savor the
m. I also put these in his <a href=" Pet Seek-A-Treat Shuffle Bone Dog
Puzzle as a reward, and he solves his puzzle with enthusiasm. Even
though my dog weighs 50 pounds, one 8 oz bag lasts about a month.

My dog has no diagnosed allergies, but his gasto-esophgial system
was sensitive and irregular starting from the time we rescued him. For
his weight-loss, I switched his food to <a href=" Buffalo Longevity Dry
Food for Adult Dogs, 24-Pound Bag, which has no soy, wheat, corn, b
y-products, or preservatives. His gastro-esophogeal system happened to
improve a lot and became regular. The hypo-allergenic treats did not co
st much more than the regular ones at the time I first bought them, so
I figured I would be safe and buy them. My dog has not had any problems
on these treats, but the change in food might also explain it. I am st
icking with the hypo-allergenic even though it is not eligible for free
shipping.

Having said that my dog likes these treats a lot,
these are not his very favorite. I give him a medium-sized biscuit <a
href=" Breath & Tartar Spearmint & Parsley Flavored Biscuits, 19.5 Ounc
e Bagevery few days, and he goes absolutely crazy for it. Alas, the
biscuits have too many calories for me to treat him more often. I would
say my dog would rate his biscuits 10 barks for taste, whereas he would
give his biscotti a solid 7 barks.

Because the biscotti trea
ts are tasty, made with wholesome ingredients, and very low in calorie
s, I rate them 5 stars. I consider liver biscotti a good investment in
my dog is health. Every dog should have liver biscotti as his or her go
-to treats.
==========
I used to be a fan of Canidae, but after $2k in recent vet bills and a
very sick dog, I found out there is a HUGE amount of Consumer Affairs c
omplaints about this company.

After blood work, ultrasound a
nd many other tests trying to figure out why my dog was not eating, was
throwing up on the rare occasion he did, had pockets of gas the size of
base balls (he is a small dog) and had all the symptoms of kidney and l
iver failure, my vet determined it might be the food. I found that hard
to believe, but we could not find anything else wrong. If you read the
complaints you will see many people have lost their dogs. They started
using one of the very companies for production whose imported food kill
ed so many animals a year or so ago.

As soon as I quit feedi

ng him this poison, he started getting better. I was one of the lucky o
nes since my pet did not die, but as I write this, he still has not gai
ned all his weight back.

The company is acting as if nothing
is wrong. Probably on the advice of their attorneys.

You do
not need to take my word for it, please read up...
==========
Green chili sauces tend to be milder than their red companions, and thi
s "Castillo Salsa Habanera - Green" is no exception. It has a nice, ti
ny little kick to it, but it is mild enough that I could just pour it s
traight into my mouth.

So, unless you do not have much toler
ance for heat, you are buying this for the flavor, not the spice. And t
his does have a nice flavor. It is a good blend of habanera, garlic, a
nd some other spices, without being too vinegary or overpowering. We h
ad some fajitas for breakfast this morning, and this Castillo Salsa was
poured quite liberally over them. Delicious.

It is not quit
e as good as my favorite green salsa,<a href=" Jalapeno Death Sauce wit
h Tequila , but a bottle of this Castillo sauce can usually be foun
d in my kitchen.
==========
QUALITY
In my opinion, having used a dozen or so brands of olive o
il, Colavita is simply the best reasonably priced olive oil which is wi
dely available. I have been using Colavita for several years now, and
have never been dissappointed. Colavita is excellent quality, dependab
le, and reasonably priced---the perfect combination for a daily general
purpose olive oil.

FLAVOR
Although not an "olive oil go
urmand", I would describe Colavita as "rich and smooth". Connoisseurs
probably have other silly terms for the flavor, such as "nutty" or "fru
ity" (but a "civilian" would never describe the flavor as nutty or frui
ty). It is not spicy like Whole Foods 360, and NOT strongly "olive-oil
y" (i.e., "herby"). Colavita is an evoo ("extra virgin olive oil"---to
us non-olive oil snobs), which means "first pressing". Most olive-oil
nuts consider the words "extra virgin" to be sacred. Practically speak
ing though, the mere label "extra virgin" is not a guarantee of quality
---most grocery store brands (evoo or not) are harsh and bitter, but so
me are good cooking oils.

PREMIUM OILS
Botique premium
oils are are as variable year to year as wines, comparably expensive, a
nd have to be bought the way you buy fine wine (that is, ideally at tas
tings, where you confirm the quality and flavor before purchase). Some
are genuinely (as in NOT-gourmet-speak) buttery and nutty---great for d

ipping. Some taste like the fragrance of flowers---great for salads.
Never cook premium olive oil---use it for dipping or drizzling, or on d
elicate salads. Colavita is a very good general purpose oil, perhaps t
he best common grocery store olive oil, but is not quite a premium oil.

Although heresy (comparable to suggesting fruit wines to a
wine connoisseur) there are inexpensive alternatives to premium botique
olive oils, including pumpkinseed, almond, sesame, and walnut oil. I p
articularly love pumpkinseed oil as a dipping oil (by itself, no vinega
r, and with a little salt). The following are examples---I can not vouc
h for the specific brands as they are different from what I use. <a hr
ef=" Pumpkinseed Oil 8.45 oz, <a href=" Almond Oil 300ml, <a hr
ef=" - Pure Sesame Oil 5.5 Oz., <a href=" Natural Walnut Oil, Cold
Pressed 16 fl oz (474 ml) Liquid

COOKING OIL
Non-ex
tra virgin olive oils (second or later pressings) are usually darker an
d stronger flavored---which adds depth and complexity to soups, sauses,
stews, chili, casseroles, bean dishes etc. I add 1/4 to 1/2 cup of oliv
e oil to such dishes. Unless you particularly want the dish to taste
"olive-oily" (I never do), cook the oil WITH the dish. However, olive
oil can "go bad"---so unless you use alot, it does not sense to use a d
ifferent cooking oil than you use on salads, for dipping etc. Colavita
is a very good general purpose oil, and therefore a good cooking oil.<b
r />
"Cooking" does NOT mean "frying" with ANY olive oil. Because
of the low smoke point (i.e., olive oil burns at a low temperature) oli
ve oil is NOT suitable for frying. That said, you can carefully fry an
egg in olive oil at a low temperature (and honestly, good fried eggs sw
imming in good olive oil are wonderful). But you can NOT fry meat nor
deep-fry anything. The best frying oil is peanut oil. Canola is a goo
d frying oil and is particularly healthy. Like olive oil there is vari
ation in the brands. Peanut oil should be slightly nutty, but not pean
utty. Canola oil should be flavorless.

DIPPING OIL
Tra
ditionally (and in the best restaurants), Italian bread, oil and balsam
ic vinegar is served as an appetizer. The diner usually mixes his own
oil and vinegar at the table. Albeit probably heresy, you can subsitut
e other premium types of vinegar (such as wine vinegar, rasberry vinega
r, or even apple vinegar) or fresh lemon juice. I usually add a few dr
ops soy sause or Braggs for saltiness and richness to the mixture. <a h
ref=" Liquid Aminos, Natural Soy Sauce Alternative, 32-Ounce Bottle ,
(Pack of 3) Colavita is a very good "everyday" dipping oil.
<b
r />SALAD OIL
The requirements for an "everyday" salad oil are a l

ittle less stringent than that for a dipping oil, because the heavy fla
vor of the vinegar, and often pamessan, bacon bits, etc. usually domina
te. However, I rarely use oil and vinegar the traditional way. For ev
eryday salads (or as a dip for raw vegies) I prefer olive-oil based may
o with a little freshly ground pepper, and some citrus zest. For speci
al salads, I use a seed or nut oil or a fruity premium olive oil (and j
ust a hint of lemon and/or orange juice and zest). Colavita is a very
good "everyday" salad oil. If you are not familiar with fresh citrus z
est (wonderful stuff), the tool to use is <a href=" 40020 Classic Zeste
r/Grater

DRIZZING OIL
A drizzling oil is an oil you
pour over a dish immediately before serving, or which you serve in a ca
rafe on the table. If you REALLY like the flavor of "ordinary good" ol
ive oil (I do not)---then you will find Colavita to be a good drizzling
oil. My hispanic wife frequently drizzles Colavita on the dishes I hav
e cooked (with Colavita), because she likes the olive-oil flavor. By t
he way, the same principle applies to the use of butter, pepper, garli
c, cheese, even soy sause---if you want the flavor to dominate, add it
just before serving; if you want the flavor to be subtle, cook the seas
oning with the dish. For most dishes I prefer a rich blend of subtle f
lavors.

HEALTHY COOKING
I go to a great deal of effort
to minimize animal fat in my diet, by very strict trimming, thorough co
oking, and by chilling broth to remove the solidified fat. However, th
e body needs fat and craves fat. The ideal solution is to replace anim
al fat with healthy vegetable oils. Olive oil is ideal for the purpos
e. Canola is better, but is essentially flavorless.

My Sout
hern grandmother could not cook any vegetables without a thick slice of
fatback (fatty bacon)---and indeed her vegetables were very tasty. Spe
aking of which, what is the point of healthy vegetables if no-one will
eat them? The solution is to add several tablespoon fulls of olive oi
l, and just a teaspoon of chipped ham <a href=" Premium Real Crumbled B
acon to the vegies. For the best flavor, cook the vegies with the
olive oil (do NOT add oil to the cooked vegetables---unless you really
like the flavor of the oil). Season as necessary with pepper and fresh
lemon juice. Truly delicious vegetables are possible---I eat them dail
y.

ECONOMY
It has been about a year since I have purcha
sed Colavita because I can buy a better but obscure brand (Bella Famigl
ia) locally for $12 per 17oz bottle. The 34 ounce tins of Colavita <a
href=" Extra Virgin Olive Oil, 34-Ounce Tins (Pack of 2) are about
half the price of the bottle per oz at this time, but is still expensiv

```
e for the quality, particularly with the shipping charge.  The tins are
small enough to use directly. But I prefer bottles, because controlling
the flow of the oil is easier (since you can see the oil approaching th
e spout), and it is easier to gauge the amount of oil I am using by eye
(how far the level drops in the glass bottle). So, I refill bottles fro
m the tins.  I have been told that olive oil stays fresher in tins (mea
sured in months and years) That becomes relevant when you do not know h
ow many months (or even years) a bottle may have sat on a grocery shel
f.<br /><br />GENERAL RECCOMENDATION<br />The price of many items, incl
uding Colavita vary wildly on Amazon, sometimes from day-to-day.  Somet
imes it is available directly from Amazon with free shipping, sometimes
not---which makes a big difference in the total coat.  After major groc
ery shopping expeditions, sit down at your computer with your reciept,
and check if you can buy any of the non-perishables through Amazon. Whe
n you find items (even if more expensive than you just paid), put the i
tem on your Amazon wish list, and add a note to the wish list of the pr
ice you just paid. Check your wish list frequently, and when you see an
item you need at a bargain price, buy it.
==================================================
```

In [29]:

```python
#remove words with numbers python: https://stackoverflow.com/a/1808237
0/4084039
#The method will remove the number with blank.
sent_179 = re.sub("\S*\d\S*", "", sent_179).strip()
print(sent_179)
```

```
These wholesome treats are only  or  calories per biscotti "crouton." M
y  pound dog has a weight problem, and these are the only treats I can
use to reward him for obedience multiple times a day.<br /><br />The bi
scotti do not crumble in my  pocket on a walk, and my dog even sniffs m
y pockets at home to ask for another one. Even though the treats are ti
ny, he does not swallow them whole. He chews them--maybe to savor them.
I also put these in his <a href=" Pet Seek-A-Treat Shuffle Bone Dog Puz
zle</a> as a reward, and he solves his puzzle with enthusiasm. Even tho
ugh my dog weighs  pounds, one  oz bag lasts about a month.<br /><br />
My dog has no diagnosed allergies, but his gasto-esophgial system was s
ensitive and irregular starting from the time we rescued him. For his w
eight-loss, I switched his food to <a href=" Buffalo Longevity Dry Food
for Adult Dogs,  Bag</a>, which has no soy, wheat, corn, by-products, o
```

```
r preservatives. His gastro-esophogeal system happened to improve a lot
and became regular. The hypo-allergenic treats did not cost much more t
han the regular ones at the time I first bought them, so I figured I wo
uld be safe and buy them. My dog has not had any problems on these trea
ts, but the change in food might also explain it.  I am sticking with t
he hypo-allergenic even though it is not eligible for free shipping.<br
/><br />Having said that my dog likes these treats a lot, these are not
his very favorite.  I give him a medium-sized biscuit <a href=" Breath
& Tartar Spearmint & Parsley Flavored Biscuits,  Ounce Bag</a>every few
days, and he goes absolutely crazy for it. Alas, the biscuits have too
many calories for me to treat him more often. I would say my dog would
rate his biscuits  barks for taste, whereas he would give his biscotti
a solid  barks.<br /><br />Because the biscotti treats are tasty, made
with wholesome ingredients, and very low in calories, I rate them  star
s.  I consider liver biscotti a good investment in my dog is health. Ev
ery dog should have liver biscotti as his or her go-to treats.
```

In [30]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
#The method will remove any special character present in the text.
sent_73 = re.sub('[^A-Za-z0-9]+', ' ', sent_179)
print(sent_73)
```

```
These wholesome treats are only or calories per biscotti crouton My pou
nd dog has a weight problem and these are the only treats I can use to
reward him for obedience multiple times a day br br The biscotti do not
crumble in my pocket on a walk and my dog even sniffs my pockets at hom
e to ask for another one Even though the treats are tiny he does not sw
allow them whole He chews them maybe to savor them I also put these in
his a href Pet Seek A Treat Shuffle Bone Dog Puzzle a as a reward and h
e solves his puzzle with enthusiasm Even though my dog weighs pounds on
e oz bag lasts about a month br br My dog has no diagnosed allergies bu
t his gasto esophgial system was sensitive and irregular starting from
the time we rescued him For his weight loss I switched his food to a hr
ef Buffalo Longevity Dry Food for Adult Dogs Bag a which has no soy whe
at corn by products or preservatives His gastro esophogeal system happe
ned to improve a lot and became regular The hypo allergenic treats did
not cost much more than the regular ones at the time I first bought the
m so I figured I would be safe and buy them My dog has not had any prob
lems on these treats but the change in food might also explain it I am
```

```
sticking with the hypo allergenic even though it is not eligible for fr
ee shipping br br Having said that my dog likes these treats a lot thes
e are not his very favorite I give him a medium sized biscuit a href Br
eath Tartar Spearmint Parsley Flavored Biscuits Ounce Bag a every few d
ays and he goes absolutely crazy for it Alas the biscuits have too many
calories for me to treat him more often I would say my dog would rate h
is biscuits barks for taste whereas he would give his biscotti a solid
barks br br Because the biscotti treats are tasty made with wholesome i
ngredients and very low in calories I rate them stars I consider liver
biscotti a good investment in my dog is health Every dog should have li
ver biscotti as his or her go to treats
```

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
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 [32]:

```
# 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%|██████████| 62862/62862 [00:27<00:00, 2276.77it/s]
```

In [33]:

```
preprocessed_reviews[179]
```

Out[33]:

```
'wholesome treats calories per biscotti crouton pound dog weight proble
m treats use reward obedience multiple times day biscotti not crumble p
ocket walk dog even sniffs pockets home ask another one even though tre
ats tiny not swallow whole chews maybe savor also put no soy wheat corn
products preservatives gastro esophogeal system happened improve lot be
came regular hypo allergenic treats not cost much regular ones time fir
st bought figured would safe buy dog not problems treats change food mi
ght also explain sticking hypo allergenic even though not eligible free
```

```
shipping said dog likes treats lot not favorite give medium sized biscu
it'
```

[3.2] Preprocessing Review Summary

In [34]:
```
#Finding the reviews which contains URLs in it.
lst = final['Summary'].values.tolist()
print(Find_Url_In_Review(lst))
#print(lst)
```

```
{60279: ['http://www.amazon.com']}
```

In [35]:
```
summ_180279 = final['Summary'].values[60279]


print(summ_180279)
```

```
http://www.amazon.com/gp/product/B007I7YYGY/ref=cm_cr_rev_prod_title
```

In [36]:
```
#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()
not in stopwords)
    preprocessed_Summary.append(Summary.strip())
```

```
100%|██████████| 62862/62862 [00:19<00:00, 3196.39it/s]
```

In [37]:
```
#Printing one random Summary
preprocessed_Summary[3435]
```

Out[37]:
```
'good til last treat'
```

In [39]:
```
final.head()
```

Out[39]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Help
22621	24751	2734888454	A1C298ITT645B6	Hugh G. Pritchard	0	0
22620	24750	2734888454	A13ISQV0U9GZIC	Sandikaye	1	1
2547	2775	B00002NCJC	A13RRPGE79XFFH	reader48	0	0
2546	2774	B00002NCJC	A196AJHU9EASJN	Alex Chaffee	0	0

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Help
1145	1244	B00002Z754	A3B8RCEI0FXFI6	B G Chase	10	10

Spliting into Train, Test and CV

In [0]:
```
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 [41]:
```
#Displaying the Final Dataframe
final.head()
```

Out[41]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Help
22621	24751	2734888454	A1C298ITT645B6	Hugh G. Pritchard	0	0

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Help
22620	24750	2734888454	A13ISQV0U9GZIC	Sandikaye	1	1
2547	2775	B00002NCJC	A13RRPGE79XFFH	reader48	0	0
2546	2774	B00002NCJC	A196AJHU9EASJN	Alex Chaffee	0	0
1145	1244	B00002Z754	A3B8RCEI0FXFI6	B G Chase	10	10

In [42]:
```
!pip install -U scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
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: scipy>=0.13.3 in /usr/
local/lib/python3.6/dist-packages (from scikit-learn) (1.1.0)
Requirement already satisfied, skipping upgrade: numpy>=1.8.2 in /usr/l
ocal/lib/python3.6/dist-packages (from scikit-learn) (1.14.6)
```

In [0]:

```
# Sorting data based on time
final["Time"] = pd.to_datetime(final["Time"], unit = "s")
final = final.sort_values(by = "Time")
```

In [47]:

```
final.tail(5)
```

Out[47]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	H
43703	47562	B004M0Y8T8	A2QJS6MHTIFSRI	Georgie	0	0
19181	20930	B001L1MKLY	A38XYFHXEUNUW6	bleaufire	0	0

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	H
30235	32932	B001P05K8Q	A3L0B5NBTQ7ZHO	Julie	0	0
6548	7178	B004OQLIHK	AKHQMSUORSA91	Pen Name	0	0
5259	5703	B009WSNWC4	AMP7K1O84DH1T	ESTY	0	0

In [49]:
```
final.count()
```

Out[49]:
```
Id                        62862
ProductId                 62862
UserId                    62862
ProfileName               62862
HelpfulnessNumerator      62862
HelpfulnessDenominator    62862
Score                     62862
Time                      62862
Summary                   62862
Text                      62862
```

```
Clean_Text                 62862
Clean_Summary              62862
Final_Text                 62862
dtype: int64
```

In [0]:
```
# split the data set into train and test
X_Temp, X_Test, Y_Temp, Y_Test = train_test_split(final['Final_Text'],
final['Score'], test_size=0.3, random_state=42)

# split the train data set into cross validation train and cross valida
tion test
X_Train, X_CV, Y_Train, Y_CV = train_test_split(X_Temp, Y_Temp, test_si
ze=0.25,random_state=42)
```

In [54]:
```
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))
print('='*50)
print("Size of X_CV :",len(X_CV))
print("Size of Y_CV :",len(Y_CV))
```

```
Size of X_Test : 18859
Size of Y_Test : 18859
==================================================
Size of X-Train : 33002
Size of Y_Train : 33002
==================================================
Size of X_CV : 11001
Size of Y_CV : 11001
```

[4] Featurization

[4.1] BAG OF WORDS

In [57]:
```
#BoW For Train
count_vect = CountVectorizer()

X_Train_BOW = count_vect.fit_transform(X_Train)

#BoW For Cross Validation
X_CV_BOW = count_vect.transform(X_CV)

#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 CV BOW vectorizer ",X_CV_BOW.get_shape())
print("the shape of out Train BOW vectorizer ",X_Test_BOW.get_shape())
```

```
the shape of out Train BOW vectorizer  (33002, 35442)
the shape of out CV BOW vectorizer  (11001, 35442)
the shape of out Train BOW vectorizer  (18859, 35442)
```

[4.2] TF-IDF

In [58]:
```
#TF_IDF For Train
tf_idf_vect = TfidfVectorizer()

X_Train_TFIDF = tf_idf_vect.fit_transform(X_Train)
#TF_IDF For CV
X_CV_TFIDF = tf_idf_vect.transform(X_CV)
#TF_IDF For Test
X_Test_TFIDF = tf_idf_vect.transform(X_Test)


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
())
print("the shape of out CV TFIDF vectorizer ",X_CV_TFIDF.get_shape())
```

```
the shape of out Train TFIDF vectorizer  (33002, 35442)
the shape of out Test TFIDF vectorizer  (18859, 35442)
the shape of out CV TFIDF vectorizer  (11001, 35442)
```

[4.4] Word2Vec

In [0]:
```
#List of words in Train
i=0
list_of_sentance_Train=[]
for sentance in X_Train:
    list_of_sentance_Train.append(sentance.split())
```

In [0]:
```
#List of words in Test
i=0
list_of_sentance_Test=[]
for sentance in X_Test:
    list_of_sentance_Test.append(sentance.split())
```

In [0]:
```
#List of word in CV
i=0
list_of_sentance_CV=[]
for sentance in X_CV:
    list_of_sentance_CV.append(sentance.split())
```

In [76]:
```
# 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 val
ues
# To use this code-snippet, download "GoogleNews-vectors-negative300.bi
n"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edi
t
```

```
# it's 1.9GB in size.


# http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17
SRFAzZPY
# 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_Train,min_count=5,size=50, work
ers=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('https://drive.google.com/file/d/0B7XkCwpI5KDYNlN
UTTlSS21pQmM/edit'):
        w2v_model=KeyedVectors.load_word2vec_format('https://drive.goog
le.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit', 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_trai
n_w2v = True, to train your own w2v ")
```

```
[('awesome', 0.8403745889663696), ('fantastic', 0.8204253315925598),
('terrific', 0.7862483859062195), ('excellent', 0.7753687500953674),
('good', 0.7698863744735718), ('wonderful', 0.7529803514480591), ('amaz
ing', 0.7493711709976196), ('perfect', 0.7159906029701233), ('nice', 0.
6633667945861816), ('outstanding', 0.663290798664093)]
==================================================
[('nastiest', 0.7660232782363892), ('greatest', 0.741354763507843), ('b
est', 0.697220504283905), ('experienced', 0.6648867726325989), ('smooth
est', 0.6601447463035583), ('disgusting', 0.6365408897399902), ('horrib
```

```
le', 0.6320235133171082), ('met', 0.617220401763916), ('weakest', 0.615
6582832336426), ('tastiest', 0.6131817698478699)]
```

In [77]:
```
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  11460
sample words  ['whole', 'family', 'loved', 'cereal', 'even', 'old', 'as
ked', 'seconds', 'could', 'taste', 'nuts', 'no', 'nut', 'pieces', 'comp
letely', 'blended', 'would', 'buy', 'one', 'best', 'looking', 'green',
'signature', 'drink', 'wedding', 'came', 'across', 'finish', 'recipe',
'called', 'kiwi', 'syrup', 'not', 'find', 'locally', 'store', 'amazon',
'rescue', 'monin', 'made', 'yummy', 'perfect', 'cocktail', 'smoke', 'fl
avor', 'light', 'texture', 'allowing', 'fat', 'fish']
```

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

[4.4.1.1] Avg W2v

In [78]:
```
# average Word2Vec for train
# compute average word2vec for each review.
train_vectors = []; # the avg-w2v for each sentence/review is stored in
 this list
for sent in tqdm(list_of_sentance_Train): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, yo
u 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/re
view
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
```

```
        sent_vec /= cnt_words
    train_vectors.append(sent_vec)
print(len(train_vectors))
print(len(train_vectors[0]))
```

```
100%|██████████| 33002/33002 [00:59<00:00, 567.72it/s]
```

```
33002
50
```

In [79]:

```
# average Word2Vec for Test
# compute average word2vec for each review.
test_vectors = []; # the avg-w2v for each sentence/review is stored in
 this list
for sent in tqdm(list_of_sentance_Test): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, yo
u 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/re
view
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    test_vectors.append(sent_vec)
print(len(test_vectors))
print(len(test_vectors[0]))
```

```
100%|██████████| 18859/18859 [00:34<00:00, 540.53it/s]
```

```
18859
50
```

In [80]:

```
# average Word2Vec for CV
# compute average word2vec for each review.
cv_vectors = []; # the avg-w2v for each sentence/review is stored in th
is list
```

```
for sent in tqdm(list_of_sentance_CV): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, yo
u 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/re
view
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    cv_vectors.append(sent_vec)
print(len(cv_vectors))
print(len(cv_vectors[0]))
```

```
100%|██████████| 11001/11001 [00:20<00:00, 531.50it/s]
```

```
11001
50
```

[4.4.1.2] TFIDF weighted W2v

In [0]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model = TfidfVectorizer()
tf_idf_matrix = model.fit_transform(X_Train)
# we are converting a dictionary with word as a key, and the idf as a v
alue
dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))
```

In [87]:
```
# TF-IDF weighted Word2Vec for train
tfidf_feat = model.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and ce
ll_val = tfidf

tfidf_w2v_Train = []; # the tfidf-w2v for each sentence/review is store
d in this list
```

```
row=0;
for sent in tqdm(list_of_sentance_Train): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    weight_sum =0; # num of words with a valid vector in the sentence/r
eview
    for word in sent: # for each word in a review/sentence
        if word in w2v_words and word in tfidf_feat:
            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_w2v_Train.append(sent_vec)
    row += 1
```

```
100%|██████████| 33002/33002 [10:50<00:00, 50.77it/s]
```

In [88]:

```
# TF-IDF weighted Word2Vec for test
tfidf_feat = model.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and ce
ll_val = tfidf

tfidf_w2v_Test = []; # the tfidf-w2v for each sentence/review is stored
 in this list
row=0;
for sent in tqdm(list_of_sentance_Test): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    weight_sum =0; # num of words with a valid vector in the sentence/r
eview
    for word in sent: # for each word in a review/sentence
        if word in w2v_words and word in tfidf_feat:
            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_w2v_Test.append(sent_vec)
    row += 1
```

```
100%|██████████| 18859/18859 [06:13<00:00, 50.53it/s]
```

In [89]:

```
# TF-IDF weighted Word2Vec for CV
tfidf_feat = model.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and ce
ll_val = tfidf

tfidf_w2v_CV = []; # the tfidf-w2v for each sentence/review is stored i
n this list
row=0;
for sent in tqdm(list_of_sentance_CV): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length
    weight_sum =0; # num of words with a valid vector in the sentence/r
eview
    for word in sent: # for each word in a review/sentence
        if word in w2v_words and word in tfidf_feat:
            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_w2v_CV.append(sent_vec)
    row += 1
```

```
100%|██████████| 11001/11001 [03:41<00:00, 49.74it/s]
```

[5] Assignment 3: KNN

1. **Apply Knn(brute force version) 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. **Apply Knn(kd tree version) on these feature sets**
   NOTE: sklearn implementation of kd-tree accepts only dense matrices, you need to convert the sparse matrices of CountVectorizer/TfidfVectorizer into dense matices. You can convert sparse matrices to dense using .toarray() attribute. For more information please visit this link

   - SET 5:Review text, preprocessed one converted into vectors using (BOW) but with restriction on maximum features generated.

```
count_vect = CountVectorizer(min_df=10, max_features=500)
count_vect.fit(preprocessed_reviews)
```

   - SET 6:Review text, preprocessed one converted into vectors using (TFIDF) but with restriction on maximum features generated.

```
tf_idf_vect = TfidfVectorizer(min_df=10, max_features=500)
tf_idf_vect.fit(preprocessed_reviews)
```

- SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
- SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)

3. **The hyper paramter tuning(find best K)**

   - Find the best hyper parameter which will give the maximum AUC 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. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure
   - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
   - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points

5. **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.

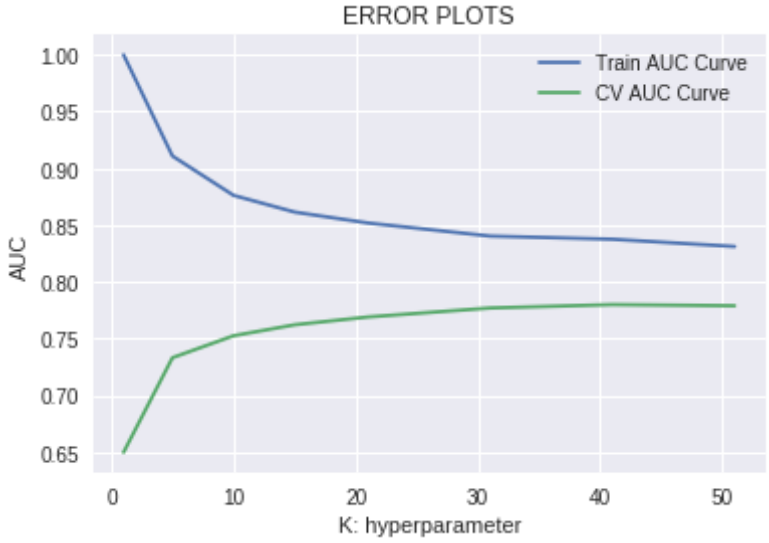
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link.

[5.1] Applying KNN brute force

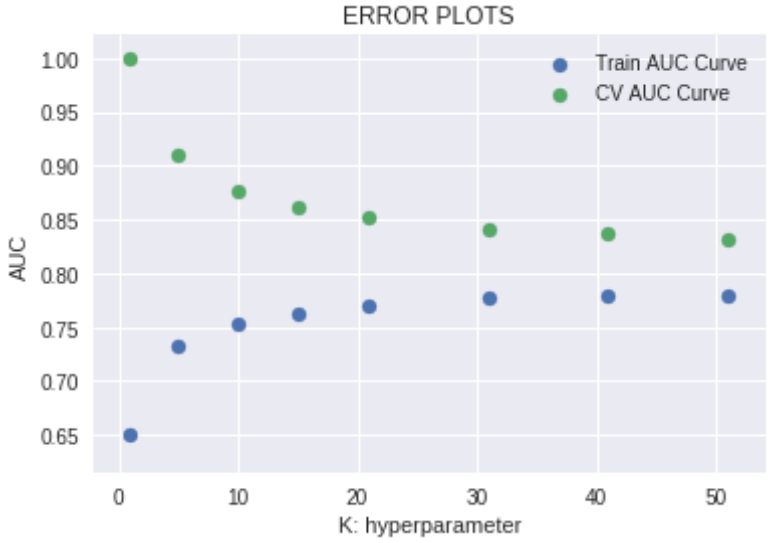
[5.1.1] Applying KNN brute force on BOW, SET 1

In [59]:
```
# Please write all the code with proper documentation
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31, 41, 51]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="brute")
  neigh.fit(X_Train_BOW, Y_Train)
  y_train_pred=neigh.predict_proba(X_Train_BOW)[:,1]
  y_cv_pred=neigh.predict_proba(X_CV_BOW)[:,1]
  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████| 8/8 [09:39<00:00, 72.88s/it]
```



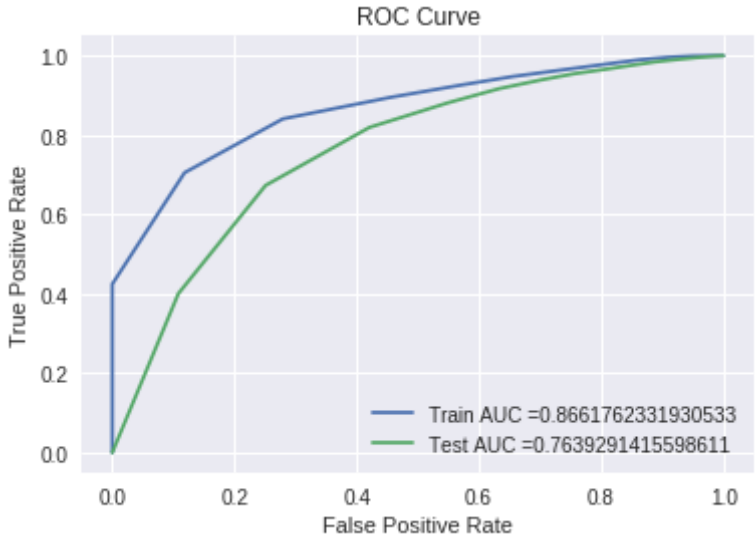
In [60]:
```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



Observation : Taking best K as 13.

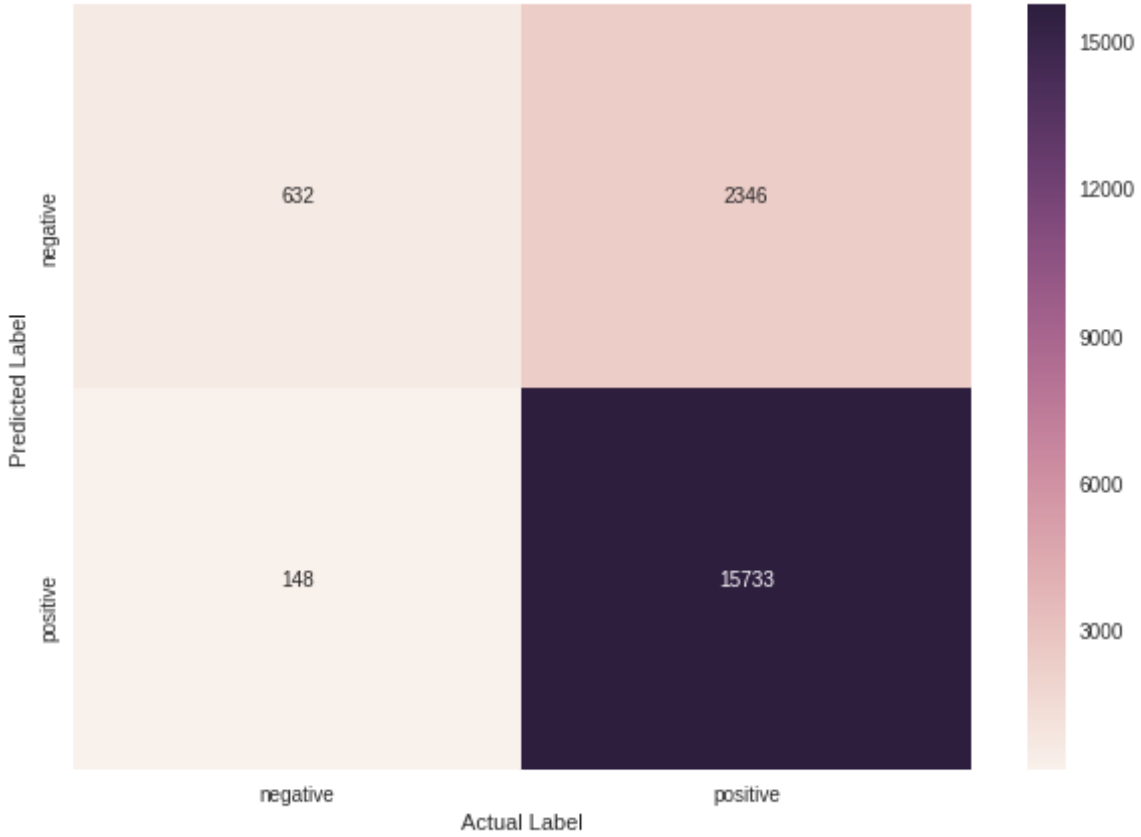
In [61]:
```
from sklearn.metrics import roc_curve, auc

optimal_k_bow_BruteForce = 13
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_bow_BruteForce,alg
orithm="brute")
neigh_bow.fit(X_Train_BOW, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(X_Train_BOW)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(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()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
plt.title("ROC Curve")
plt.show()
```



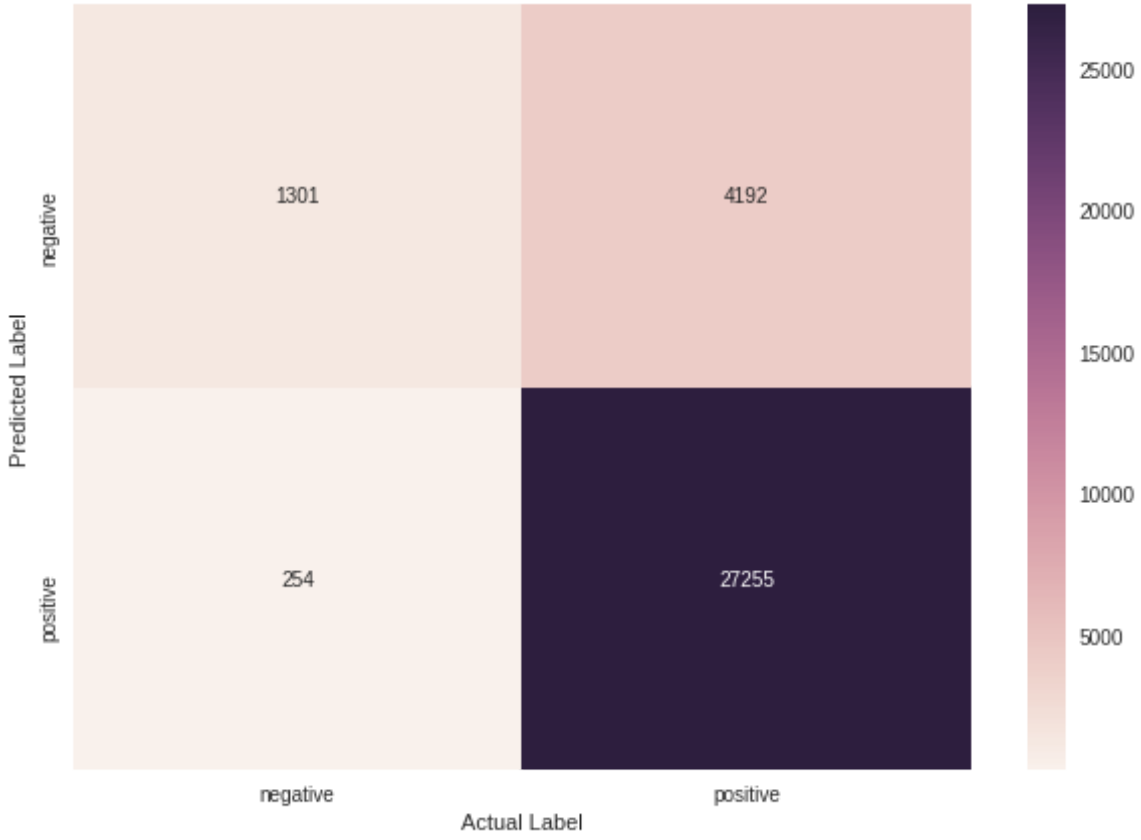
In [66]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(X_Test_BOW))

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
```



In [67]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusio
n-matrix
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.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
```



[5.1.2] Applying KNN brute force on TFIDF, SET 2

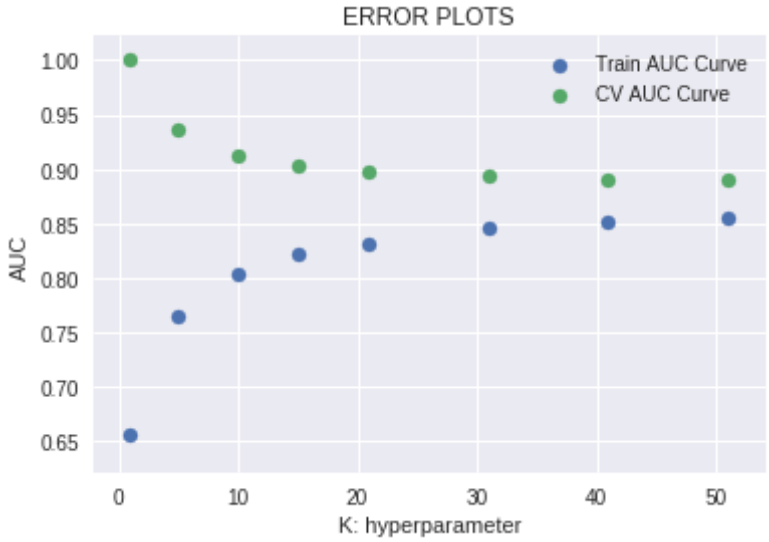
In [64]:
```
# Please write all the code with proper documentation
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31, 41, 51]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="brute")
  neigh.fit(X_Train_TFIDF, Y_Train)
```

```
  y_train_pred=neigh.predict_proba(X_Train_TFIDF)[:,1]
  y_cv_pred=neigh.predict_proba(X_CV_TFIDF)[:,1]
  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████| 8/8 [09:30<00:00, 72.02s/it]
```

In [68]:
```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
```

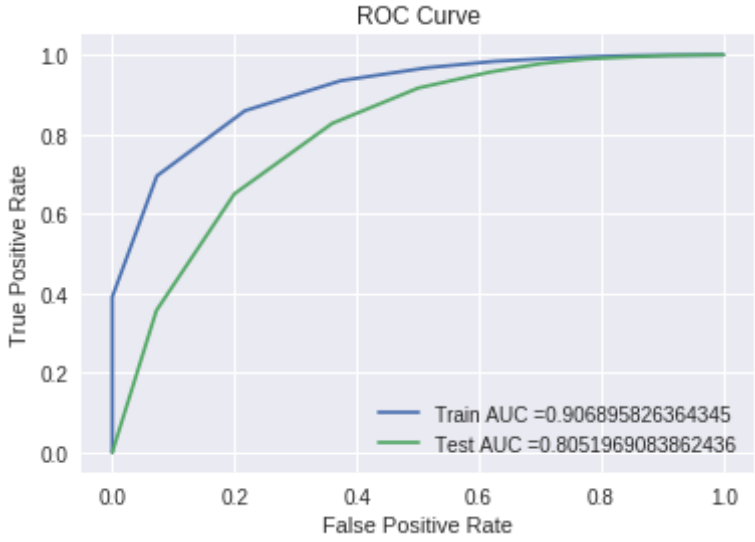
```
plt.title("ERROR PLOTS")
plt.show()
```



Taking best K as 13

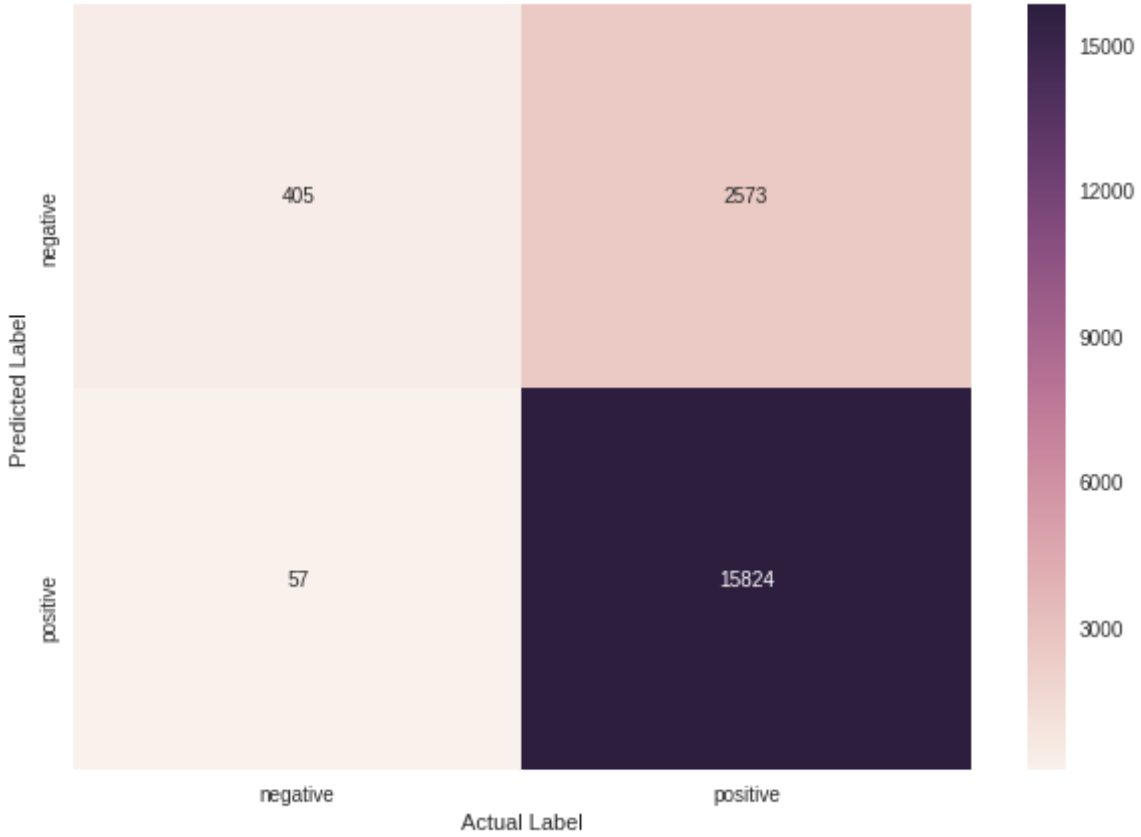
In [69]:
```
optimal_k_tfidf_BruteForce = 13
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_tfidf_BruteForce,a
lgorithm="brute")
neigh_bow.fit(X_Train_TFIDF, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(X_Train_TFIDF)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(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")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()
```



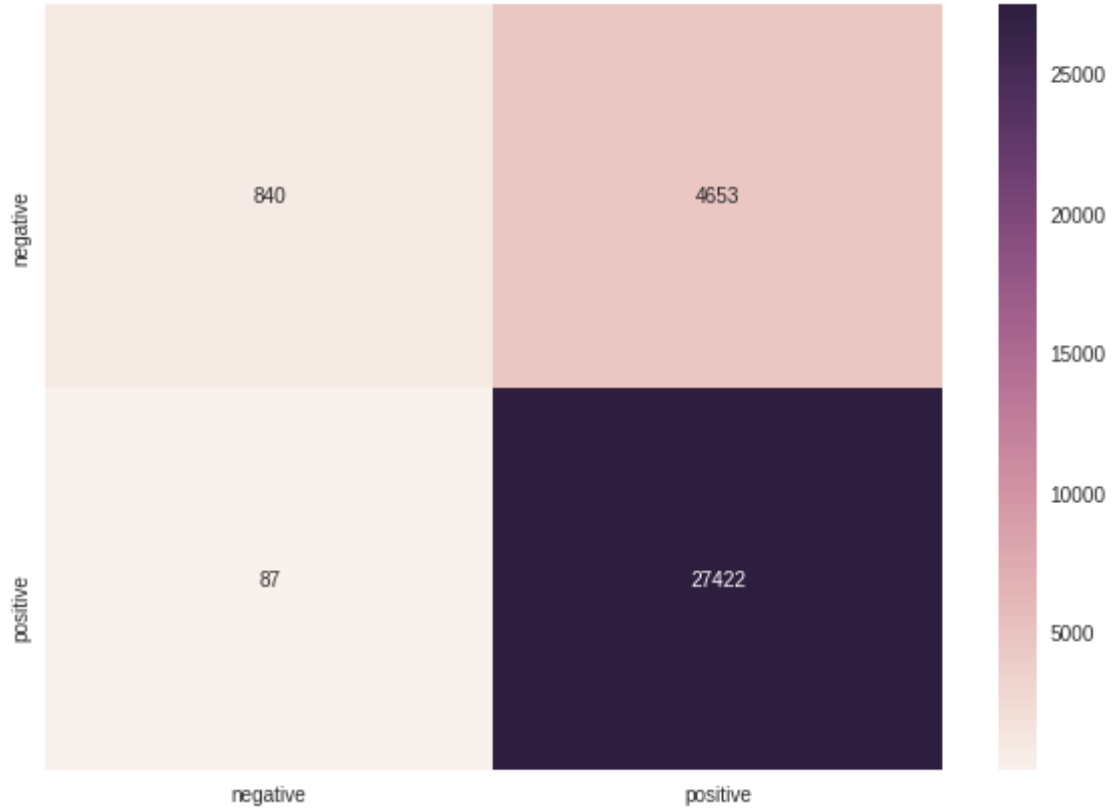
In [70]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(X_Test_TFIDF))

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
```



In [71]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.predict(X_Train_TFIDF))

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

```
Train confusion matrix
```

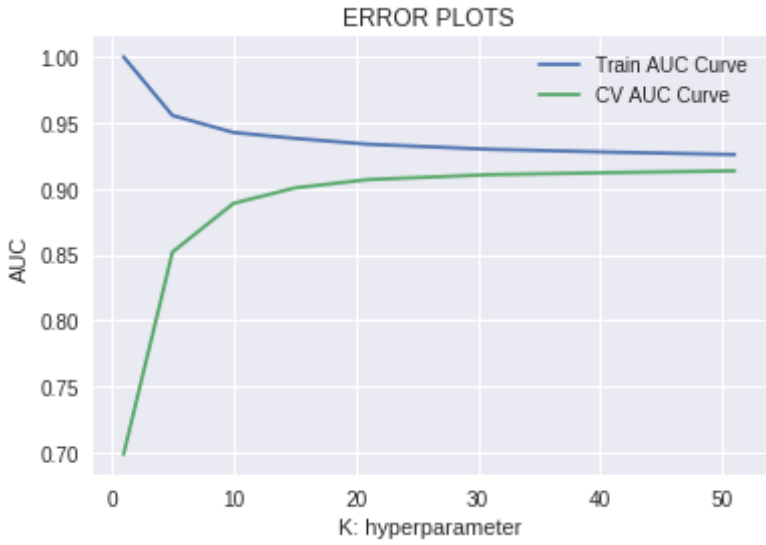


[5.1.3] Applying KNN brute force on AVG W2V, SET 3

In [81]:
```
# Please write all the code with proper documentation
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31, 41, 51]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="brute")
  neigh.fit(train_vectors, Y_Train)
  # roc_auc_score(y_true, y_score) the 2nd parameter should be probabil
ity estimates of t# not the predicted outputs
```

```
  y_train_pred=neigh.predict_proba(train_vectors)[:,1]
  y_cv_pred=neigh.predict_proba(cv_vectors)[:,1]
  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

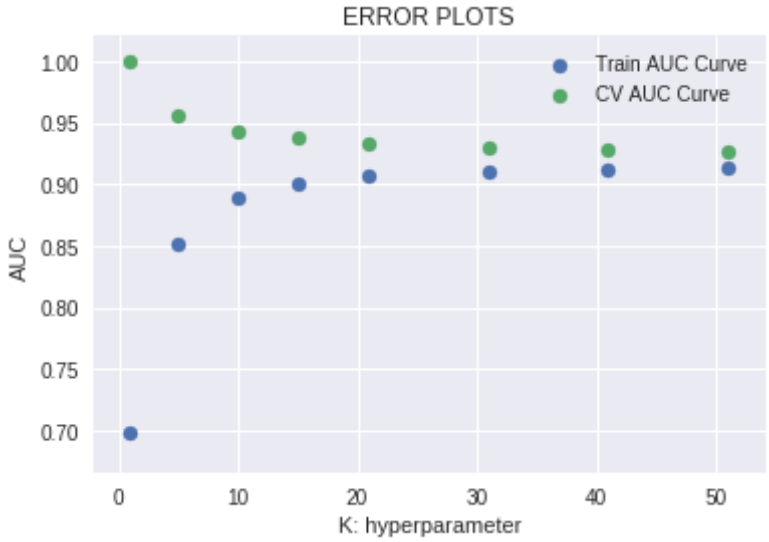
```
100%|██████████| 8/8 [04:37<00:00, 35.29s/it]
```



In [82]:

```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
```

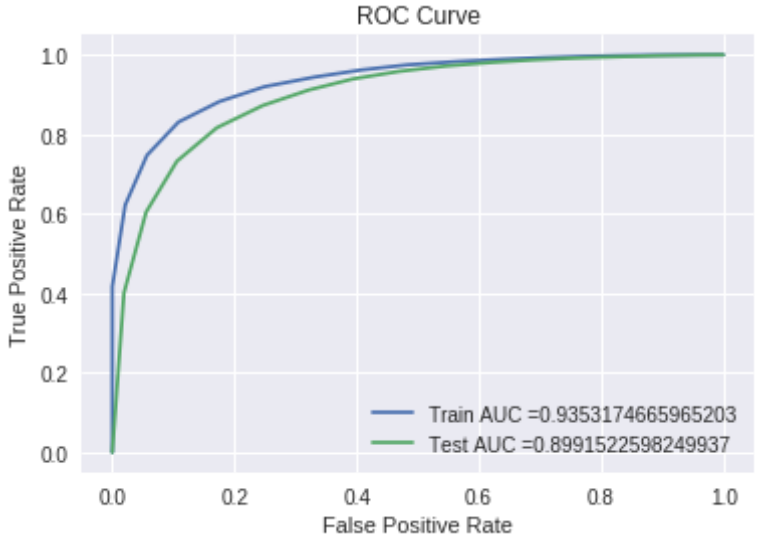
```
plt.title("ERROR PLOTS")
plt.show()
```



Taking best K as 19

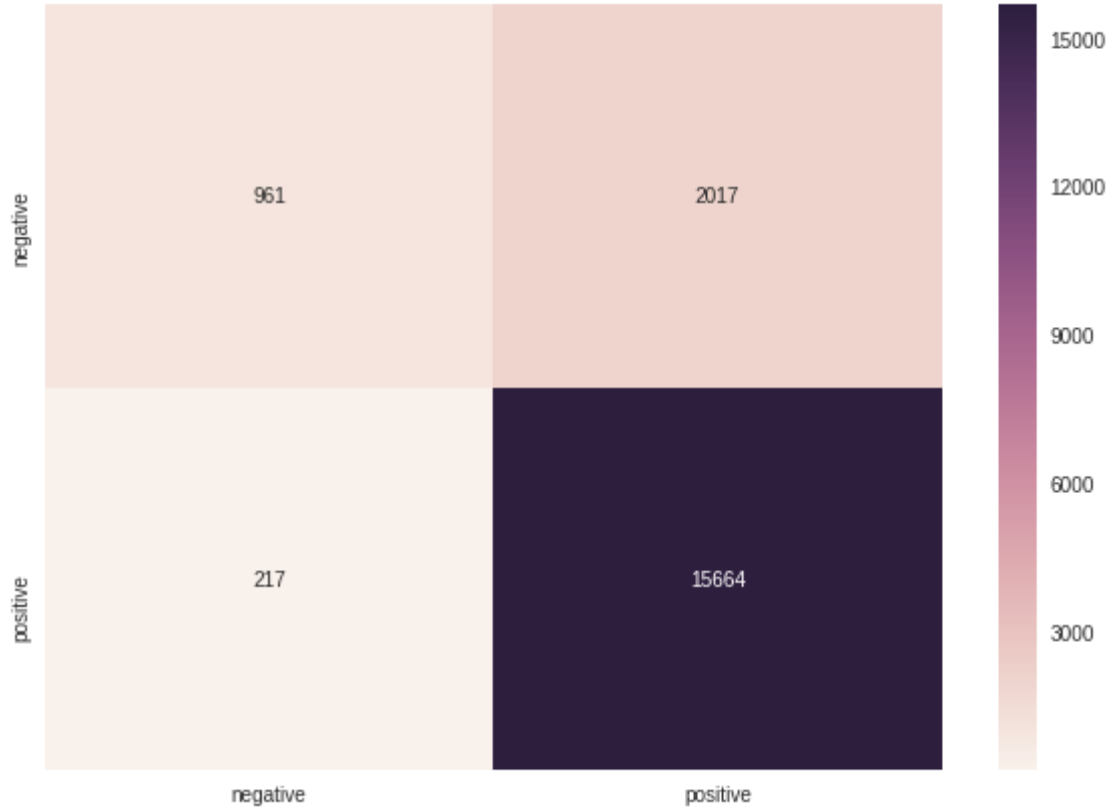
In [83]:
```
optimal_k_avgw2vec_BruteForce = 19
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_avgw2vec_BruteForc
e,algorithm="brute")
neigh_bow.fit(train_vectors, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(train_vectors)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(test_vectors )[:,1])
train_avgw2vec_acc =auc(train_fpr, train_tpr)
test_avgw2vec_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(train_avgw2vec_a
cc))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(test_avgw2vec_acc))
plt.legend()
```

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



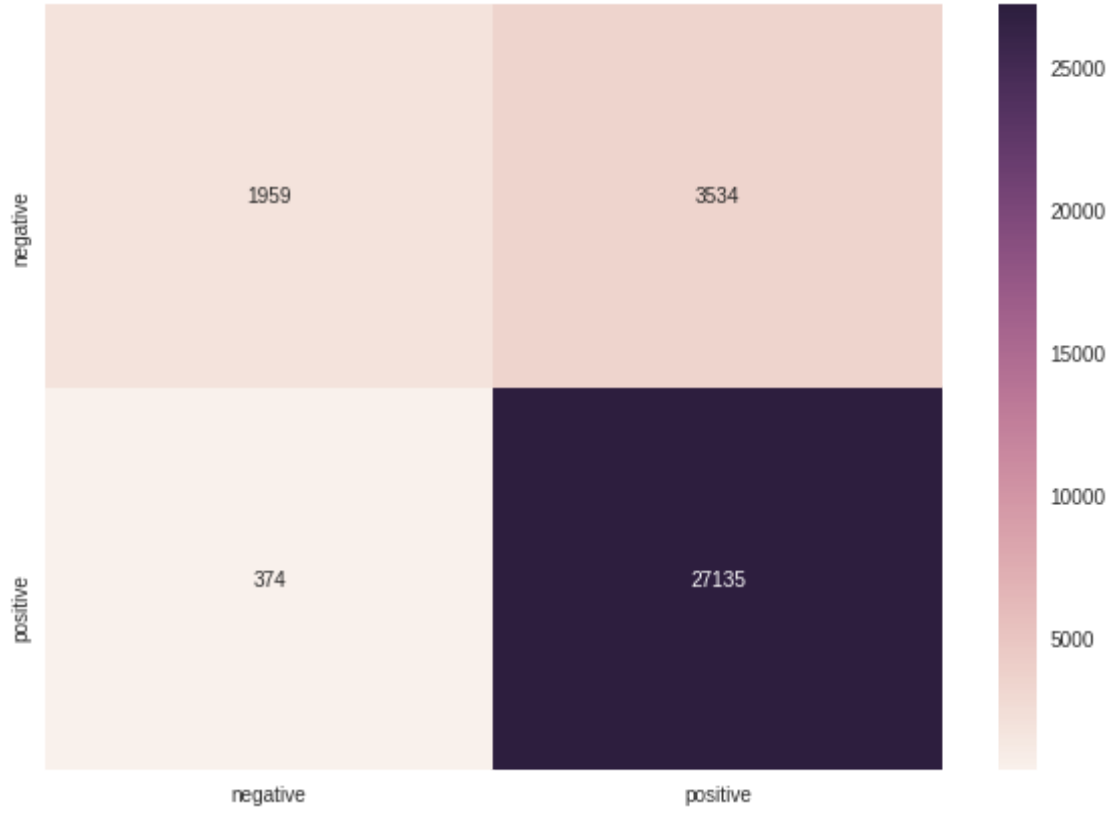
In [84]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(test_vectors))

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

```
Test confusion matrix
```



In [85]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.predict(train_vectors))

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

```
Train confusion matrix
```



[5.1.4] Applying KNN brute force on TFIDF W2V, SET 4

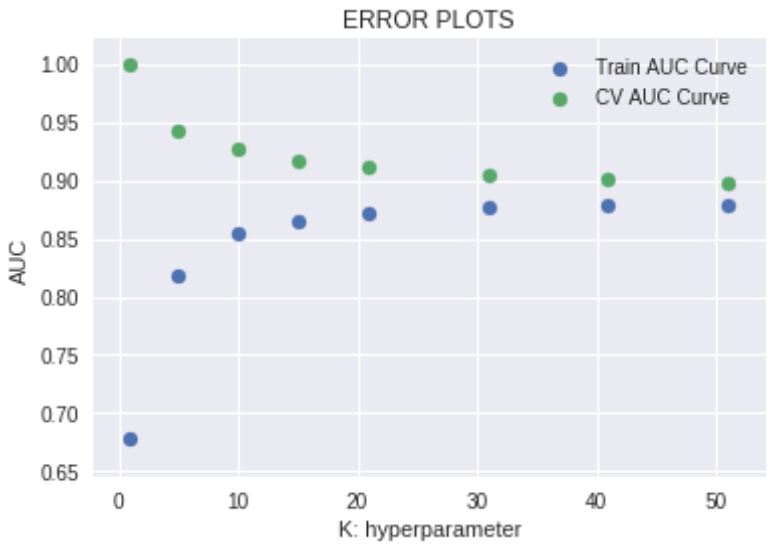
In [90]:
```
# Please write all the code with proper documentation
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31, 41, 51]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="brute")
  neigh.fit(tfidf_w2v_Train, Y_Train)

  y_train_pred=neigh.predict_proba(tfidf_w2v_Train)[:,1]
```

```
  y_cv_pred=neigh.predict_proba(tfidf_w2v_CV)[:,1]

  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████| 8/8 [04:29<00:00, 34.27s/it]
```

In [91]:
```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
```

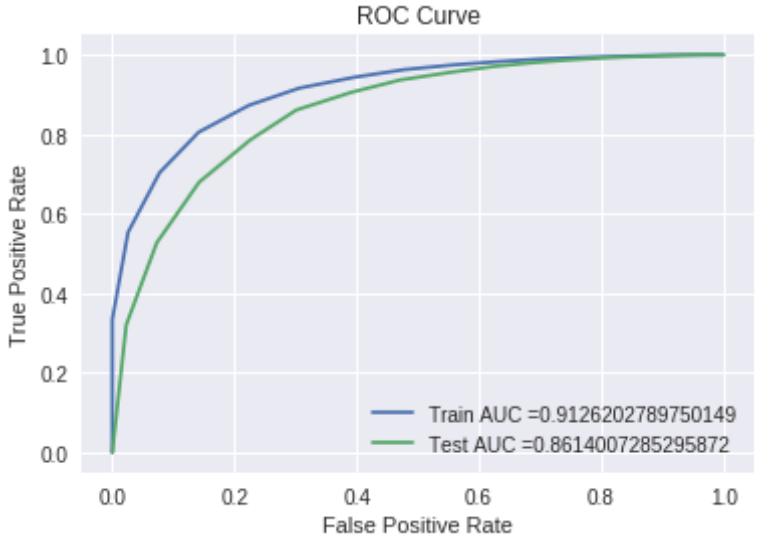
```
plt.title("ERROR PLOTS")
plt.show()
```



Best K as 19

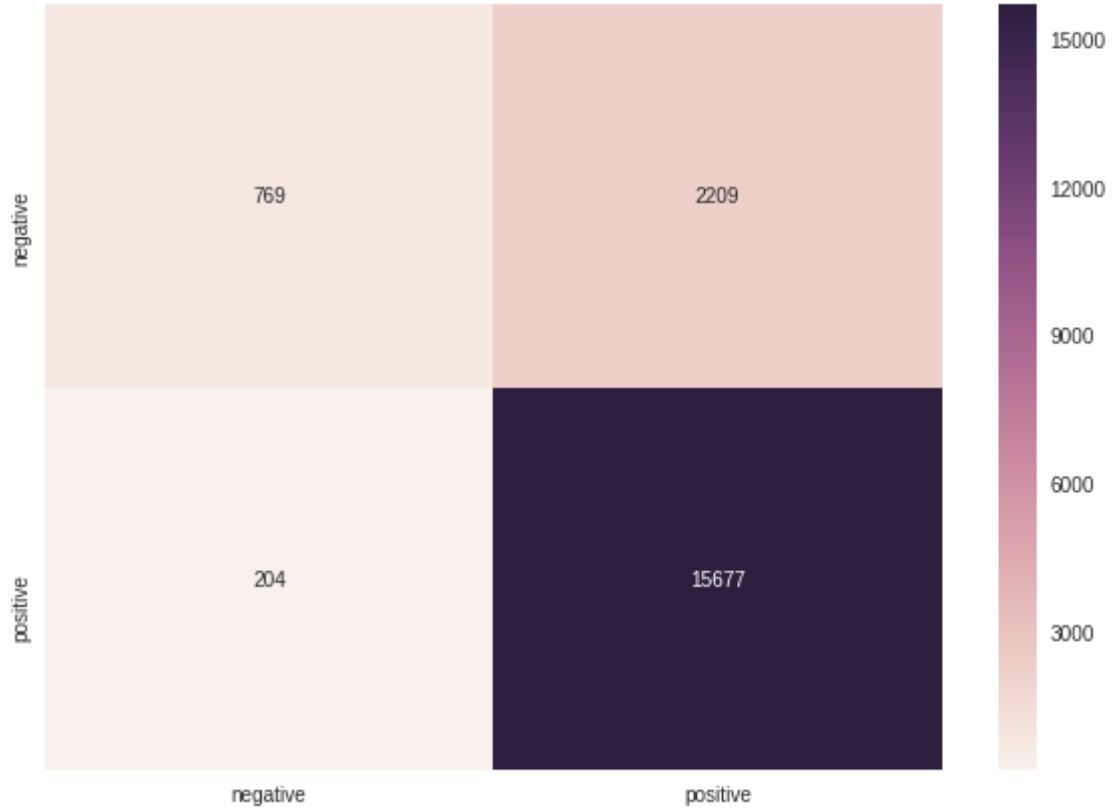
In [92]:
```
optimal_k_tfidfw2v_BruteForce = 19
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_tfidfw2v_BruteForc
e,algorithm="brute")
neigh_bow.fit(tfidf_w2v_Train, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(tfidf_w2v_Train)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(tfidf_w2v_Test )[:,1])
train_tfidfw2v_acc =auc(train_fpr, train_tpr)
test_tfidfw2v_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(train_tfidfw2v_a
cc))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(test_tfidfw2v_acc))
plt.legend()
```

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



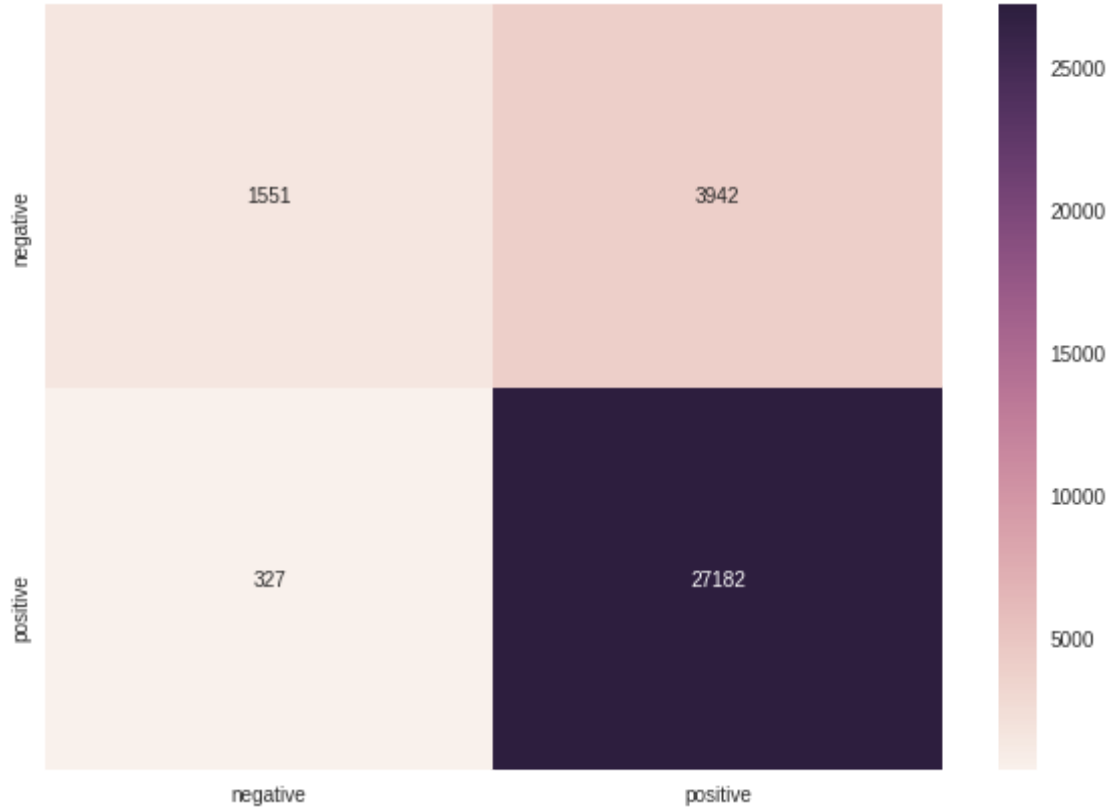
In [93]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(tfidf_w2v_Test))

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

```
Test confusion matrix
```



In [95]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.predict(tfidf_w2v_Train))

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

```
Train confusion matrix
```



[5.2] Applying KNN kd-tree

[5.2.1] Applying KNN kd-tree on BOW, SET 5

In [0]:
```
# Please write all the code with proper documentation
from sklearn.decomposition import TruncatedSVD
count_vect=CountVectorizer(min_df=10, max_features=500)

X_Train_BOW = count_vect.fit_transform(X_Train)
X_Test_BOW = count_vect.transform(X_Test)
X_CV_BOW = count_vect.transform(X_CV)
```

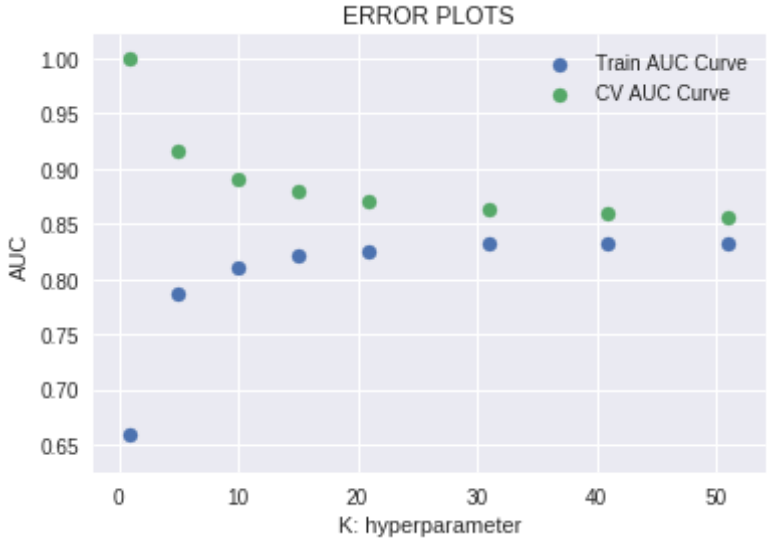
```
svd=TruncatedSVD(n_components=100)
#Convertin into Dense Vector
x_train_dense_BOW = svd.fit_transform(X_Train_BOW)
x_test_dense_BOW = svd.transform(X_Test_BOW)
x_cv_dense_BOW = svd.transform(X_CV_BOW)
```

In [97]:

```
# Applying KNN using KD-Tree
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31, 41, 51]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="kd_tree")
  neigh.fit(x_train_dense_BOW, Y_Train)
  # roc_auc_score(y_true, y_score) the 2nd parameter should be probabil
ity estimates of t# not the predicted outputs
  y_train_pred=neigh.predict_proba(x_train_dense_BOW)[:,1]
  y_cv_pred=neigh.predict_proba(x_cv_dense_BOW)[:,1]
  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOT")
plt.show()
```

```
100%|██████████| 8/8 [37:28<00:00, 296.30s/it]
```

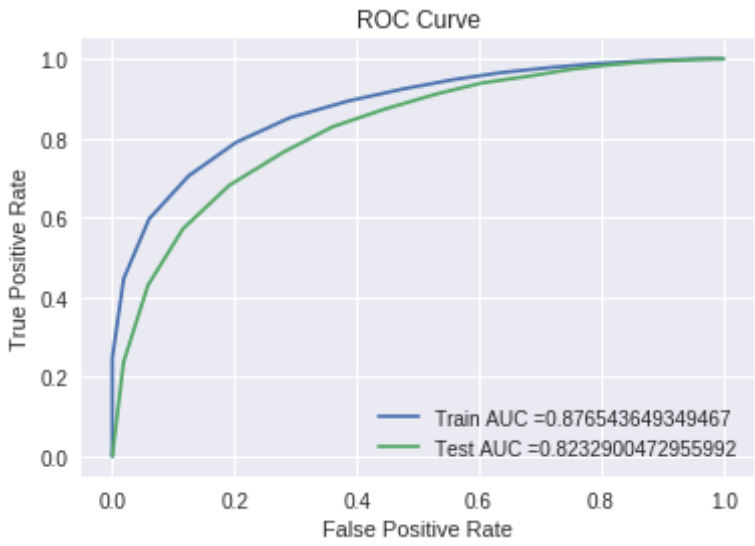
In [98]:
```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



Taking best K as 17

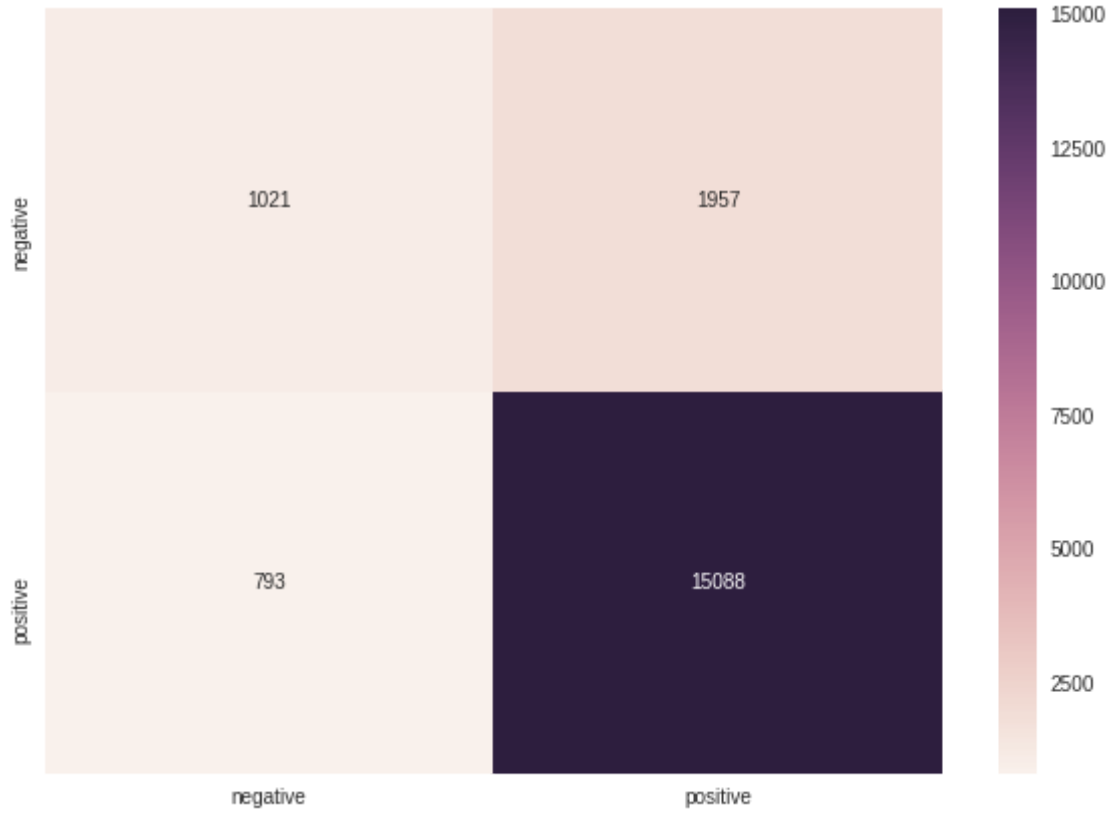
In [99]:
```
optimal_k_bow_KdTree = 17
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_bow_KdTree,algorit
hm="kd_tree")
neigh_bow.fit(x_train_dense_BOW, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(x_train_dense_BOW)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(x_test_dense_BOW )[:,1])
train_bow_kd_acc =auc(train_fpr, train_tpr)
test_bow_kd_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(train_bow_kd_acc
))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(test_bow_kd_acc))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
plt.title("ROC Curve")
plt.show()
```



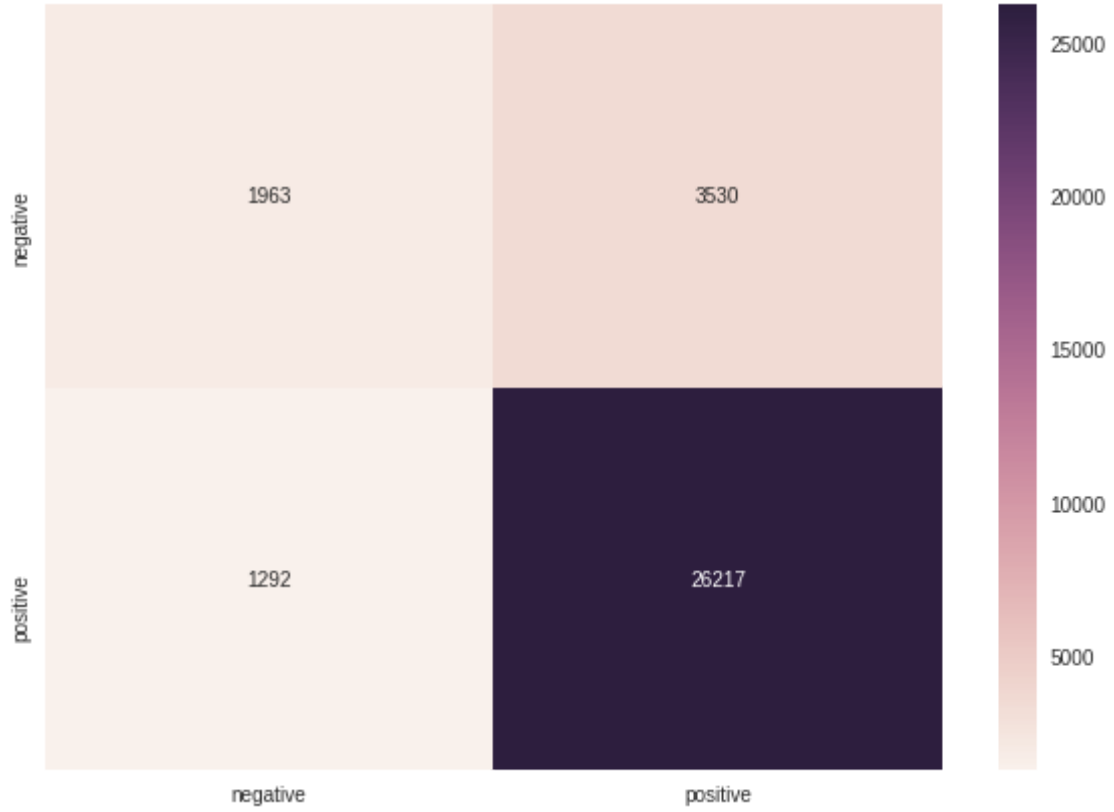
In [100]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(x_test_dense_BOW))

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

```
Test confusion matrix
```



In [101]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.predict(x_train_dense_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))
plt.xlabel("Actual Label")
plt.ylabel("Predicted Label")
sns.heatmap(df_cm, annot=True,fmt="d")
plt.show()
```

```
Train confusion matrix
```



[5.2.2] Applying KNN kd-tree on TFIDF, SET 6

In [0]:
```
# Please write all the code with proper documentation
from sklearn.decomposition import TruncatedSVD
count_vect=CountVectorizer(min_df=10, max_features=500)

X_Train_TFIDF = count_vect.fit_transform(X_Train)
X_Test_TFIDF = count_vect.transform(X_Test)
X_CV_TFIDF = count_vect.transform(X_CV)

svd=TruncatedSVD(n_components=100)
#Convertin into Dense Vector
```

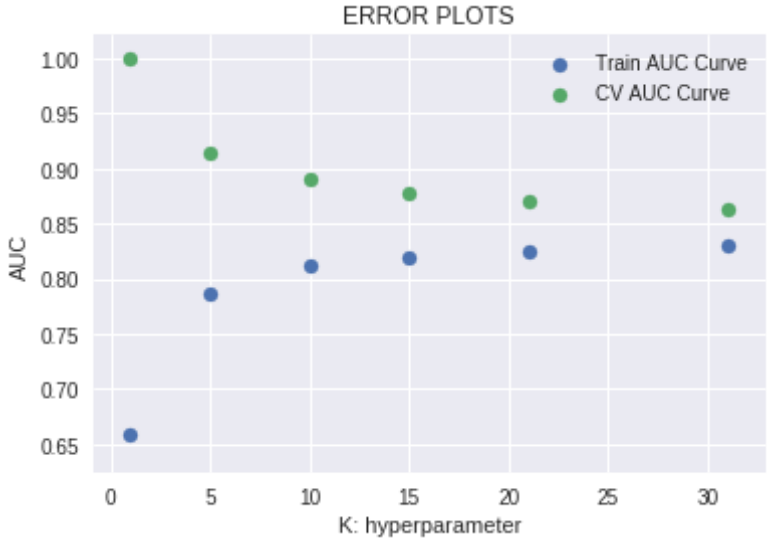
```
x_train_dense_TFIDF = svd.fit_transform(X_Train_TFIDF)
x_test_dense_TFIDF = svd.transform(X_Test_TFIDF)
x_cv_dense_TFIDF = svd.transform(X_CV_TFIDF)
```

In [103]:

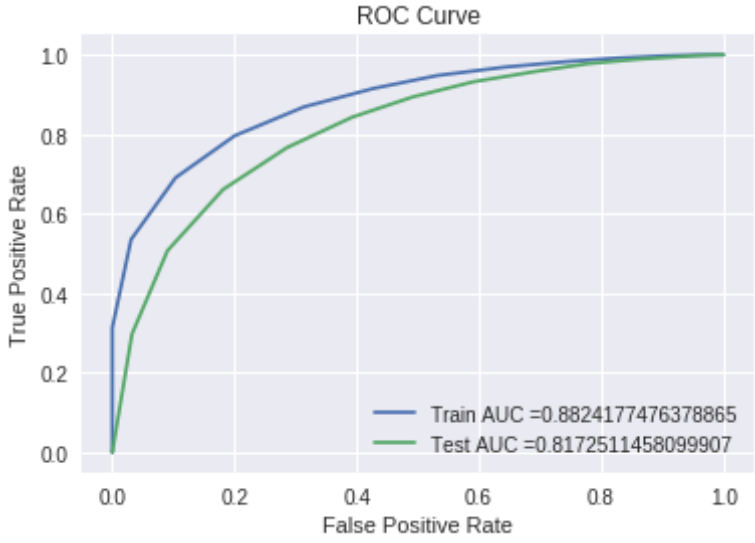
```
# Applying KNN using KD-Tree
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="kd_tree")
  neigh.fit(x_train_dense_TFIDF, Y_Train)

  y_train_pred=neigh.predict_proba(x_train_dense_TFIDF)[:,1]
  y_cv_pred=neigh.predict_proba(x_cv_dense_TFIDF)[:,1]
  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOT")
plt.show()
```

```
100%|██████████| 6/6 [26:36<00:00, 268.96s/it]
```

In [104]:
```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

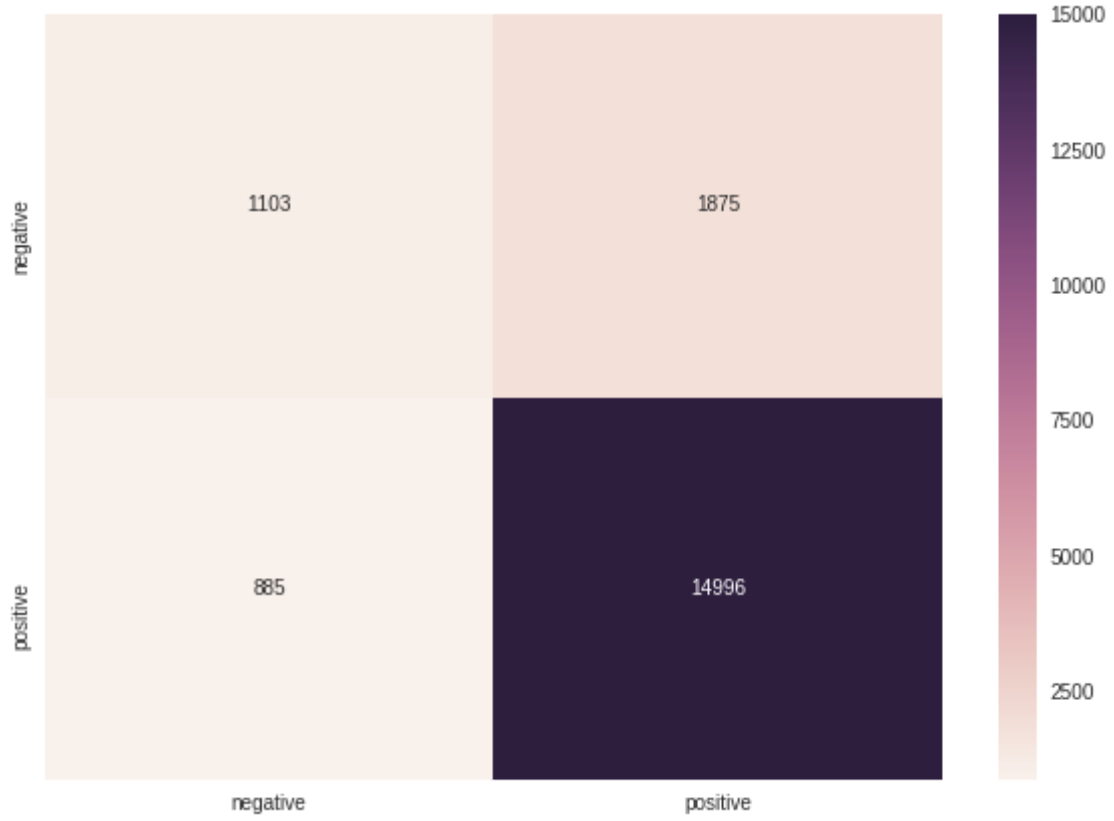


In [107]:
```
optimal_k_tfidf_KdTree = 13
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_tfidf_KdTree,algor
ithm="kd_tree")
neigh_bow.fit(x_train_dense_TFIDF, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(x_train_dense_TFIDF)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(x_test_dense_TFIDF )[:,1])
train_tfidf_kd_acc =auc(train_fpr, train_tpr)
test_tfidf_kd_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(train_tfidf_kd_a
cc))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(test_tfidf_kd_acc))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()
```



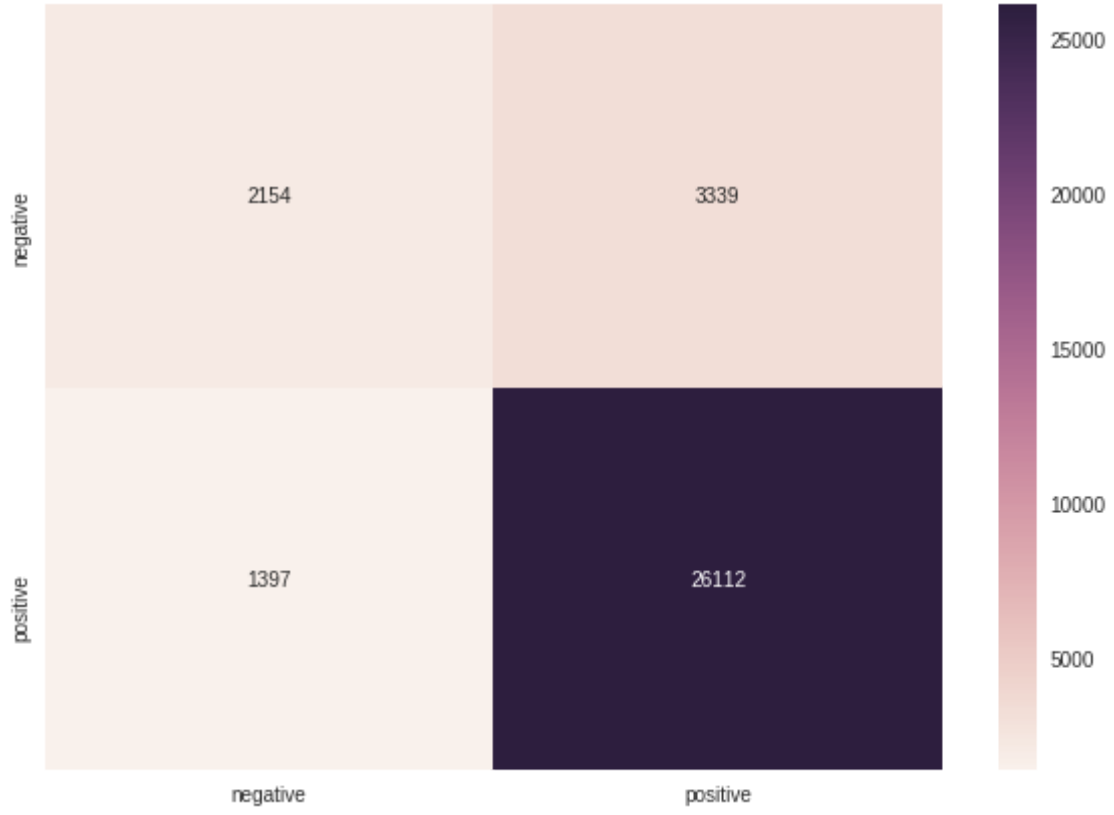
In [108]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(x_test_dense_TFIDF))

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

```
Test confusion matrix
```



In [109]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.predict(x_train_dense_TFIDF))

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

```
Train confusion matrix
```



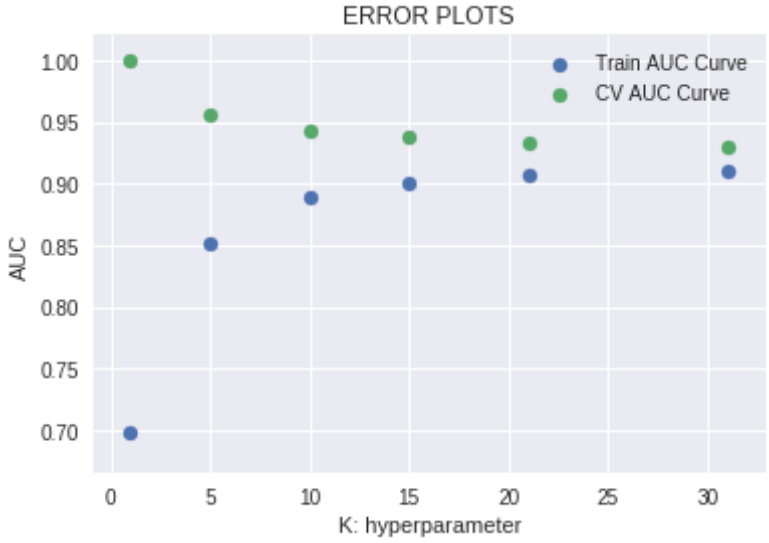
[5.2.3] Applying KNN kd-tree on AVG W2V, SET 3

In [110]:
```
# Please write all the code with proper documentation
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="kd_tree")
  neigh.fit(train_vectors, Y_Train)
  y_train_pred=neigh.predict_proba(train_vectors)[:,1]
  y_cv_pred=neigh.predict_proba(cv_vectors)[:,1]
```

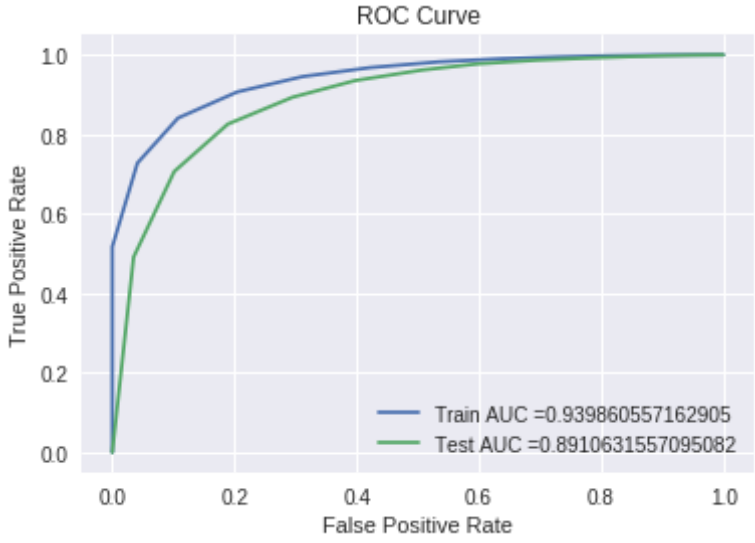
```
  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████| 6/6 [14:13<00:00, 144.82s/it]
```

In [111]:
```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

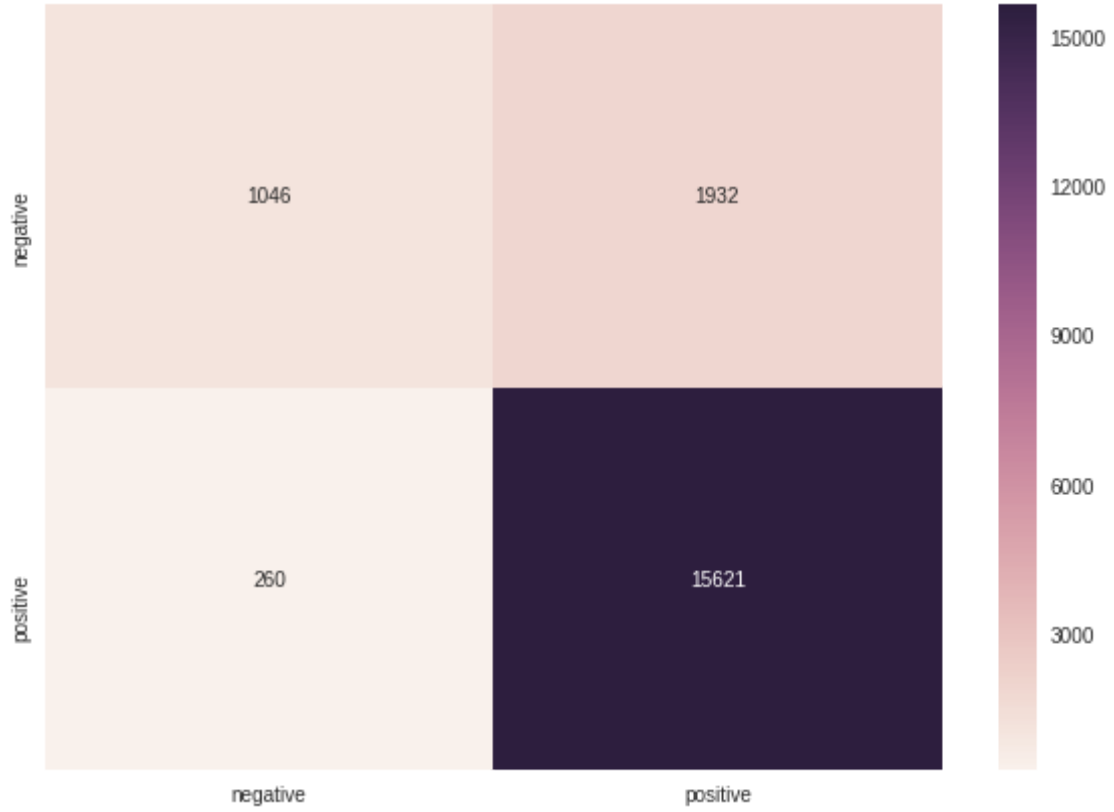


In [112]:
```
optimal_k_avgw2v_KdTree = 13
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_avgw2v_KdTree,algo
rithm="kd_tree")
neigh_bow.fit(train_vectors, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(train_vectors)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(test_vectors )[:,1])
train_avgw2v_kd_acc =auc(train_fpr, train_tpr)
test_avgw2v_kd_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(train_avgw2v_kd_
acc))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(test_avgw2v_kd_acc
))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()
```



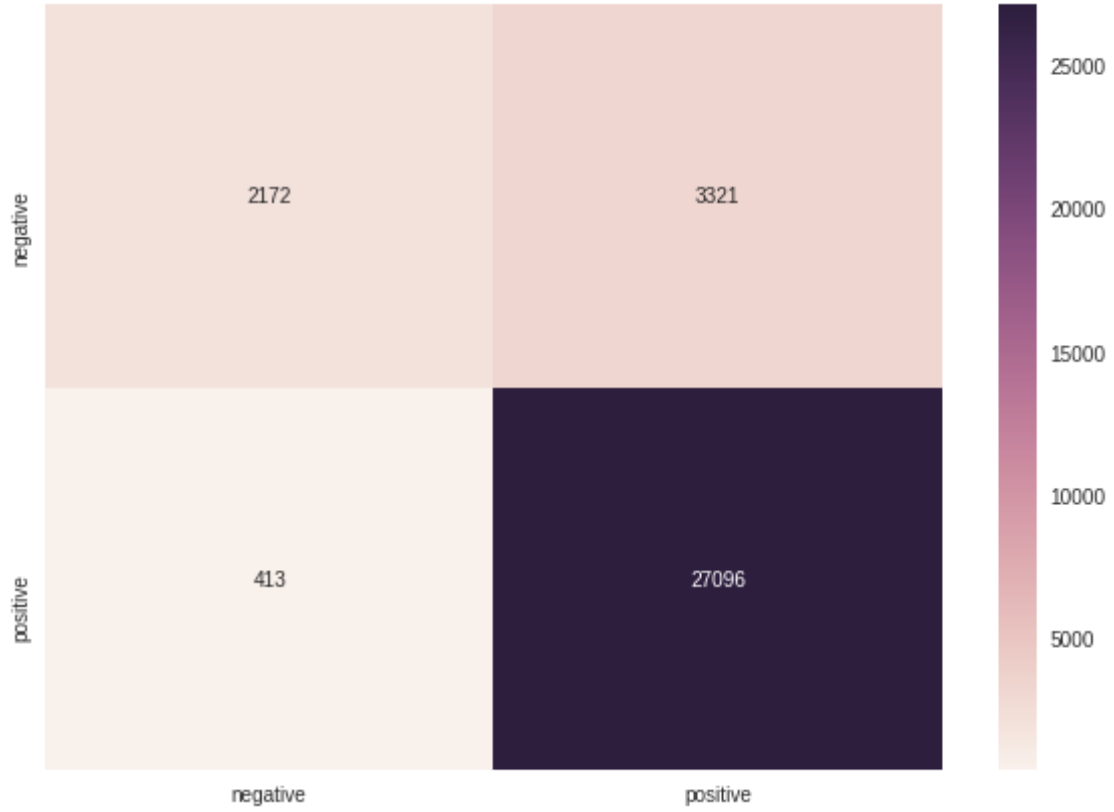
In [113]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(test_vectors))

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

```
Test confusion matrix
```



In [115]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.predict(train_vectors))

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

```
Train confusion matrix
```

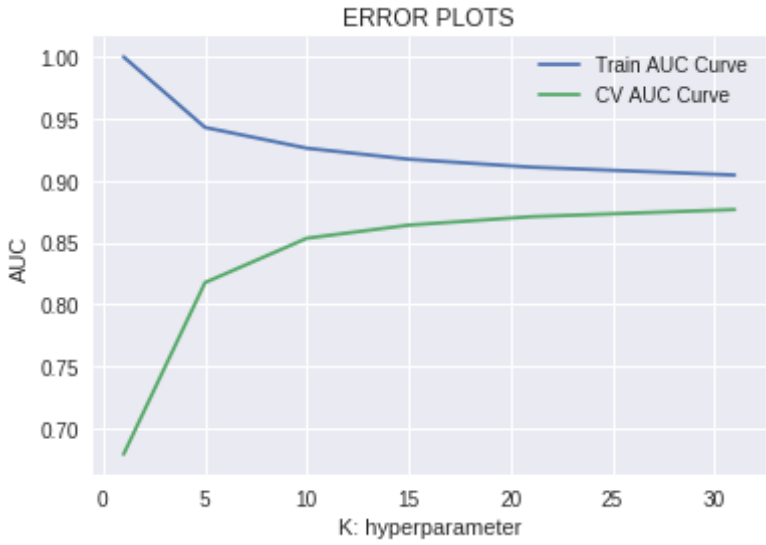


[5.2.4] Applying KNN kd-tree on TFIDF W2V, SET 4

In [116]:
```
# Please write all the code with proper documentation
from sklearn.metrics import roc_auc_score
train_auc= []
cv_auc= []
K = [1, 5, 10, 15, 21, 31]
for i in tqdm(K):
  neigh=KNeighborsClassifier(n_neighbors=i,algorithm="kd_tree")
  neigh.fit(tfidf_w2v_Train, Y_Train)

  y_train_pred=neigh.predict_proba(tfidf_w2v_Train)[:,1]
```

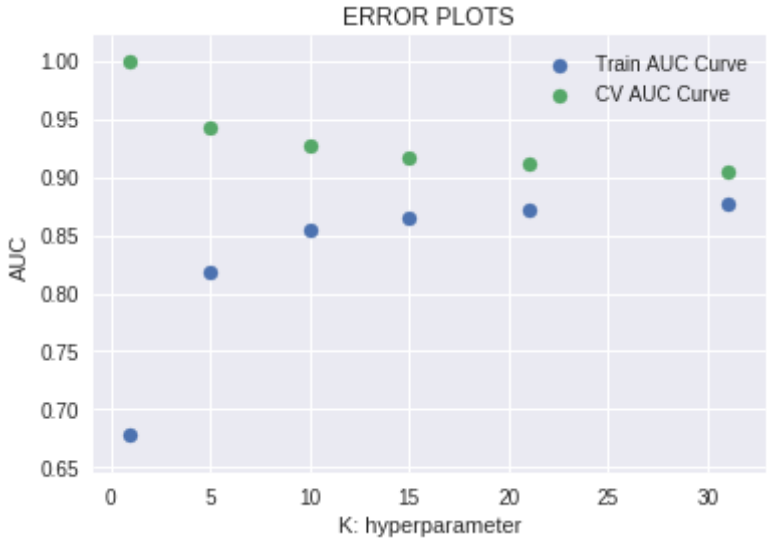
```
  y_cv_pred=neigh.predict_proba(tfidf_w2v_CV)[:,1]

  train_auc.append(roc_auc_score(Y_Train,y_train_pred))
  cv_auc.append(roc_auc_score(Y_CV, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC Curve')
plt.plot(K, cv_auc, label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████| 6/6 [11:36<00:00, 119.12s/it]
```



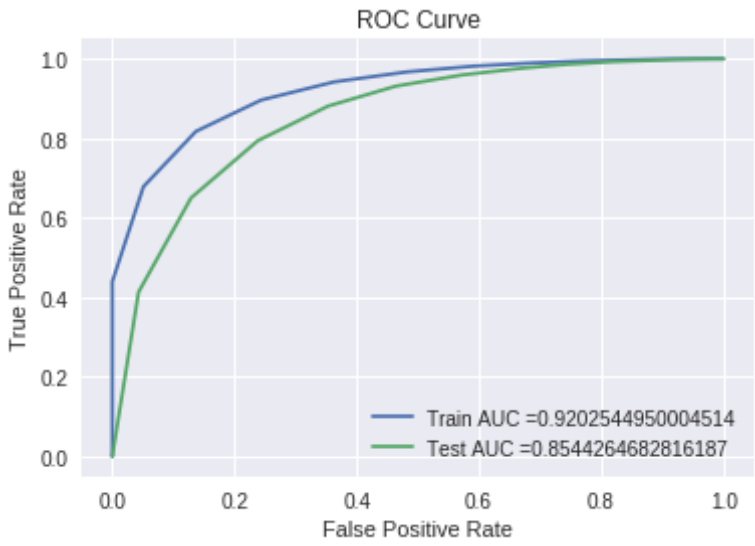
In [117]:
```
# Distribution of K values
plt.scatter(K,cv_auc,label='Train AUC Curve')
plt.scatter(K,train_auc,label='CV AUC Curve')
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
```

```
plt.title("ERROR PLOTS")
plt.show()
```



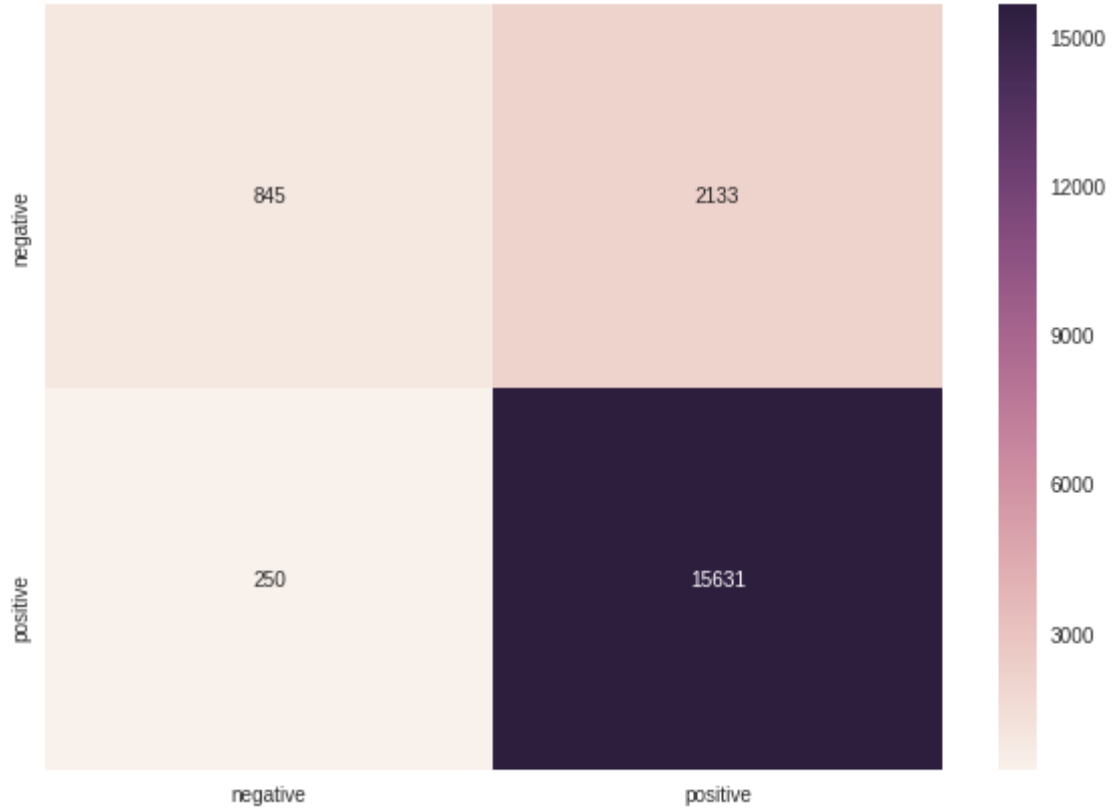
In [118]:
```
optimal_k_tfidf_avgw2v_KdTree = 13
neigh_bow=KNeighborsClassifier(n_neighbors=optimal_k_tfidf_avgw2v_KdTre
e,algorithm="kd_tree")
neigh_bow.fit(tfidf_w2v_Train, Y_Train)

train_fpr,train_tpr,thresholds = roc_curve(Y_Train, neigh_bow.predict_p
roba(tfidf_w2v_Train)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(Y_Test,  neigh_bow.predict_p
roba(tfidf_w2v_Test )[:,1])
train_tfidfw2v_kd_acc =auc(train_fpr, train_tpr)
test_tfidfw2v_kd_acc = auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(train_tfidfw2v_k
d_acc))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(test_tfidfw2v_kd_ac
c))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
```

```
plt.title("ROC Curve")
plt.show()
```



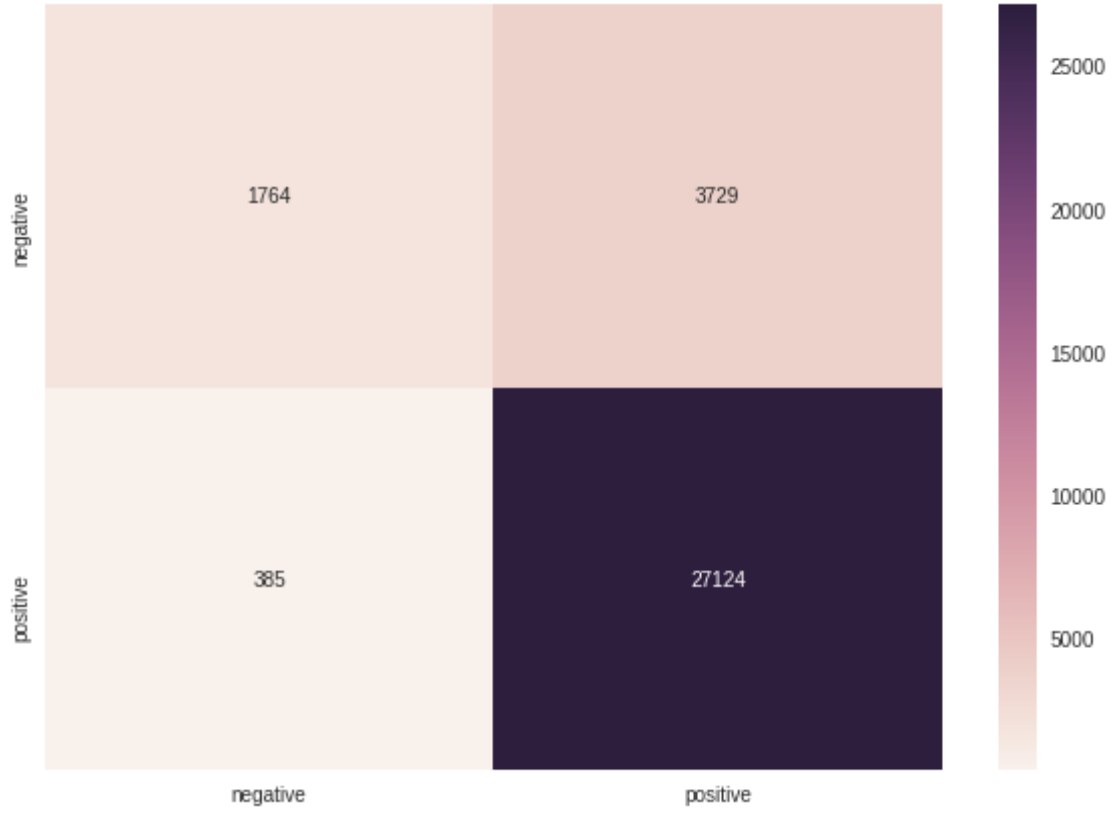
In [119]:
```
class_names= ['negative','positive']
print("Test confusion matrix")
array = confusion_matrix(Y_Test, neigh.predict(tfidf_w2v_Test))

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

```
Test confusion matrix
```



In [120]:
```
class_names= ['negative','positive']
print("Train confusion matrix")
array = confusion_matrix(Y_Train, neigh.predict(tfidf_w2v_Train))

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

```
Train confusion matrix
```



[6] Conclusions

In [121]:
```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

names= [
        "KNN using 'brute' for BoW",
                    "KNN using 'brute' for TFIDF",
                    "KNN using 'brute' for Avg-Word2Vec",
                    "KNN using 'brute' for TFIDF-Word2Vec",
                    "KNN using 'kdTree' for BoW",
                    "KNN using 'kdTree' for TFIDF",
```

```
                    "KNN using 'kdTree' for Avg-Word2Vec",
                    "KNN using 'kdTree' for TFIDF-Word2Vec"
                ]


optimal_K= [
                        optimal_k_bow_BruteForce,
                        optimal_k_tfidf_BruteForce,
                        optimal_k_avgw2vec_BruteForce,
                        optimal_k_tfidfw2v_BruteForce,
                        optimal_k_bow_KdTree,
                        optimal_k_tfidf_KdTree,
                        optimal_k_avgw2v_KdTree,
                        optimal_k_tfidf_avgw2v_KdTree
                        ]

train_acc= [
                        train_bow_acc,
                        train_tfidf_acc,
                        train_avgw2vec_acc,
                        train_tfidfw2v_acc,
                        train_bow_kd_acc,
                        train_tfidf_kd_acc,
                        train_avgw2v_kd_acc,
                        train_tfidfw2v_kd_acc
                        ]

test_acc = [
                        test_bow_acc,
                        test_tfidf_acc,
                        test_avgw2vec_acc,
                        test_tfidfw2v_acc,
                        test_bow_kd_acc,
                        test_tfidf_kd_acc,
                        test_avgw2v_kd_acc,
                        test_tfidfw2v_kd_acc
 ]
numbering= [1,2,3,4,5,6,7,8]
```

```
# Initializing prettytable

ptable=PrettyTable()

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

# Printing the Table
print(ptable)
```

```
+-------+-------------------------------------+--------+-------------
-------+--------------------+
| S.NO. |                 MODEL                 | Best K | Training Acc
uracy  |   Test Accuracy    |
+-------+-------------------------------------+--------+-------------
-------+--------------------+
|   1   |       KNN using 'brute' for BoW       |   13   | 0.8661762331
930533 | 0.7639291415598611 |
|   2   |      KNN using 'brute' for TFIDF      |   13   | 0.9068958263
64345  | 0.8051969083862436 |
|   3   |   KNN using 'brute' for Avg-Word2Vec  |   19   | 0.9353174665
965203 | 0.8991522598249937 |
|   4   |  KNN using 'brute' for TFIDF-Word2Vec |   19   | 0.9126202789
750149 | 0.8614007285295872 |
|   5   |       KNN using 'kdTree' for BoW      |   17   | 0.8765436493
49467  | 0.8232900472955992 |
|   6   |      KNN using 'kdTree' for TFIDF     |   13   | 0.8824177476
378865 | 0.8172511458099907 |
|   7   |  KNN using 'kdTree' for Avg-Word2Vec  |   13   | 0.9398605571
62905  | 0.8910631557095082 |
|   8   | KNN using 'kdTree' for TFIDF-Word2Vec |   13   | 0.9202544950
004514 | 0.8544264682816187 |
+-------+-------------------------------------+--------+-------------
-------+--------------------+
```

Conclusion :

1. I have taken 70000 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, CV and Test.
6. Applied BOW, TFIDF, AVGW2V and TFIDF weighted W2V on review text.
7. Applied KNN using brute force on BOW,TFIDF, AVGW2V and TFIDF weighted W2V.
8. Implemented confusion matrix and AOC to find out accuracy.
9. Applied KNN using KD-Tree on BOW,TFIDF, AVGW2V and TFIDF weighted W2V
10. Implemented confusion matrix and AOC to find out accuracy.
11. Represent the best K, Training Accuracy and Test Accuracy in a taburar form.