DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Descr
`project_id`	A unique identifier for the proposed project. **Example:** `p0`

Feature	Descr
project_title	Title of the project. **Exar** • Art Will Make You Ha • First Grade
project_grade_category	Grade level of students for which the project is targeted. One of the fol enumerated v • Grades Pr • Grades • Grades • Grades
project_subject_categories	One or more (comma-separated) subject categories for the project fr following enumerated list of v • Applied Lear • Care & Hu • Health & Sp • History & Ci • Literacy & Lang • Math & Sci • Music & The • Special N • Wa **Exam** • Music & The • Literacy & Language, Math & Sci
school_state	State where school is located (Two-letter U.S. posta (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_c **Exampl**

Feature	Descr
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the p **Exam** • `Lite` • `Literature & Writing, Social Scie`
`project_resource_summary`	An explanation of the resources needed for the project. **Exa** • `My students need hands on literacy materials to ma sensory needs!</`
`project_essay_1`	First application
`project_essay_2`	Second application
`project_essay_3`	Third application
`project_essay_4`	Fourth application
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-0 12:43:56`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Exa** `bdf8baa8fedef6bfeec7ae4ff1c1`
`teacher_prefix`	Teacher's title. One of the following enumerated va • • • • • • `Teac`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same te **Examp**

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_3:** "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

  For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [75]:

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

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

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

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

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

from tqdm import tqdm
import os

from plotly import plotly
```

```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.1 Reading Data

In [76]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [77]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [78]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[78]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [79]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of `project_subject_subcategories`

In [80]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 Text preprocessing

In [81]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [82]:

```
project_data.head(2)
```

Out[82]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datet
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22

In [83]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [84]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third lan
guages. We are a melting pot of refugees, immigrants, and native-born Americans bringing
the gift of language to our school. \r\n\r\n We have over 24 languages represented in our
English Learner program with students at every level of mastery.  We also have over 40 co
untries represented with the families within our school.  Each student brings a wealth of
knowledge and experiences to us that open our eyes to new cultures, beliefs, and respec
t.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Our
English learner's have a strong support system at home that begs for more resources.  Man
y times our parents are learning to read and speak English along side of their children.
Sometimes this creates barriers for parents to be able to help their child learn phonetic
s, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and play
ers, students are able to continue their mastery of the English language even if no one a
t home is able to assist.  All families with students within the Level 1 proficiency stat
us, will be a offered to be a part of this program.  These educational videos will be spe
cially chosen by the English Learner Teacher and will be sent home regularly to watch.  T
he videos are to help the child develop early reading skills.\r\n\r\nParents that do not
have access to a dvd player will have the opportunity to check out a dvd player to use fo
r the year.  The plan is to use these videos and educational dvd's for the years to come
for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learn
ing, at least most of the time. At our school, 97.3% of the students receive free or redu
ced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school has a v
```

```
ibrant community that loves to get together and celebrate. Around Halloween there is a wh
ole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo
we put on a big festival with crafts made by the students, dances, and games. At the end
of the year the school hosts a carnival to celebrate the hard work put in during the scho
ol year, with a dunk tank being the most popular activity.My students will use these five
brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will
only have a total of ten in the classroom and not enough for each student to have an indi
vidual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in
place of chairs at my small group tables during math and reading times. The rest of the d
ay they will be used by the students who need the highest amount of movement in their lif
e in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is missin
g, my students always say more Hokki Stools. They can't get their fill of the 5 stools we
already have. When the students are sitting in group with me on the Hokki Stools, they ar
e always moving, but at the same time doing their work. Anytime the students get to pick
where they can sit, the Hokki Stools are the first to be taken. There are always students
who head over to the kidney table to get one of the stools who are disappointed as there
are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Ho
kki stools will be a compromise that allow my students to do desk work and move at the sa
me time. These stools will help students to meet their 60 minutes a day of movement by al
lowing them to activate their core muscles for balance while they sit. For many of my stu
dents, these chairs will take away the barrier that exists in schools for a child who ca
n't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain wall
s, rows of desks, and a teacher in front of the room? A typical day in our room is nothin
g like that. I work hard to create a warm inviting themed room for my students look forwa
rd to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and gi
rls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a h
igh enough percentage of free and reduced-price lunch to qualify. Our school is an \"open
classroom\" concept, which is very unique as there are no walls separating the classroom
s. These 9 and 10 year-old students are very eager learners; they are like sponges, absor
bing all the information and experiences and keep on wanting more.With these resources su
ch as the comfy red throw pillows and the whimsical nautical hanging decor and the blue f
ish nets, I will be able to help create the mood in our classroom setting to be one of a
themed nautical environment. Creating a classroom environment is very important in the su
```

```
ccess in each and every child's education. The nautical photo props will be used with eac
h child as they step foot into our classroom for the first time on Meet the Teacher eveni
ng. I'll take pictures of each child with them, have them developed, and then hung in our
classroom ready for their first day of 4th grade.  This kind gesture will set the tone be
fore even the first day of school! The nautical thank you cards will be used throughout t
he year by the students as they create thank you cards to their team groups.\r\n\r\nYour
generous donations will help me to help make our classroom a fun, inviting, learning envi
ronment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to
get our classroom ready. Please consider helping with this project to make our new school
year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delay
s, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and alway
s strive to work their hardest working past their limitations. \r\n\r\nThe materials we h
ave are the ones I seek out for my students. I teach in a Title I school where most of th
e students receive free or reduced price lunch.  Despite their disabilities and limitatio
ns, my students love coming to school and come eager to learn and explore.Have you ever f
elt like you had ants in your pants and you needed to groove and move as you were in a me
eting? This is how my kids feel all the time. The want to be able to move as they learn o
r so they say.Wobble chairs are the answer and I love then because they develop their cor
e, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn
through games, my kids don't want to sit and do worksheets. They want to learn to count b
y jumping and playing. Physical engagement is the key to our success. The number toss and
color and shape mats can make that happen. My students will forget they are doing work an
d just have the fun a 6 year old deserves.nannan
==================================================

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates.
The great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is m
akeup is 97.6% African-American, making up the largest segment of the student body. A typ
ical school in Dallas is made up of 23.2% African-American students. Most of the students
are on free or reduced lunch. We aren't receiving doctors, lawyers, or engineers children
from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young chil
dren and we focus not only on academics but one smart, effective, efficient, and discipli
ned students with good character.In our classroom we can utilize the Bluetooth for swift
transitions during class. I use a speaker which doesn't amplify the sound enough to recei
```

```
ve the message. Due to the volume of my speaker my students can't hear videos or books cl
early and it isn't making the lessons as meaningful. But with the bluetooth speaker my st
udents will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart
will allow me to have more room for storage of things that are needed for the day and has
an extra part to it I can use.  The table top chart has all of the letter, words and pict
ures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [85]:

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

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

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

In [86]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delay
s, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and alway
s strive to work their hardest working past their limitations. \r\n\r\nThe materials we h
ave are the ones I seek out for my students. I teach in a Title I school where most of th
e students receive free or reduced price lunch.  Despite their disabilities and limitatio
ns, my students love coming to school and come eager to learn and explore.Have you ever f
elt like you had ants in your pants and you needed to groove and move as you were in a me
eting? This is how my kids feel all the time. The want to be able to move as they learn o
r so they say.Wobble chairs are the answer and I love then because they develop their cor
e, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn
through games, my kids do not want to sit and do worksheets. They want to learn to count
by jumping and playing. Physical engagement is the key to our success. The number toss an
d color and shape mats can make that happen. My students will forget they are doing work
and just have the fun a 6 year old deserves.nannan
==================================================
```

In [87]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delay
s, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and alway
s strive to work their hardest working past their limitations.     The materials we have
are the ones I seek out for my students. I teach in a Title I school where most of the st
udents receive free or reduced price lunch.  Despite their disabilities and limitations,
my students love coming to school and come eager to learn and explore.Have you ever felt
like you had ants in your pants and you needed to groove and move as you were in a meetin
g? This is how my kids feel all the time. The want to be able to move as they learn or so
they say.Wobble chairs are the answer and I love then because they develop their core, wh
ich enhances gross motor and in Turn fine motor skills.   They also want to learn through
games, my kids do not want to sit and do worksheets. They want to learn to count by jumpi
ng and playing. Physical engagement is the key to our success. The number toss and color
and shape mats can make that happen. My students will forget they are doing work and just
have the fun a 6 year old deserves.nannan
```

In [88]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays
cognitive delays gross fine motor delays to autism They are eager beavers and always stri
ve to work their hardest working past their limitations The materials we have are the one
s I seek out for my students I teach in a Title I school where most of the students recei
ve free or reduced price lunch Despite their disabilities and limitations my students lov
e coming to school and come eager to learn and explore Have you ever felt like you had an
ts in your pants and you needed to groove and move as you were in a meeting This is how m
y kids feel all the time The want to be able to move as they learn or so they say Wobble
chairs are the answer and I love then because they develop their core which enhances gros
s motor and in Turn fine motor skills They also want to learn through games my kids do no
t want to sit and do worksheets They want to learn to count by jumping and playing Physic
al engagement is the key to our success The number toss and color and shape mats can make
that happen My students will forget they are doing work and just have the fun a 6 year ol
d deserves nannan
```

In [89]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"]
```

In [90]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████| 109248/109
248 [01:25<00:00, 1283.01it/s]
```

In [91]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[91]:
```
'my kindergarten students varied disabilities ranging speech language delays cognitive de
lays gross fine motor delays autism they eager beavers always strive work hardest working
past limitations the materials ones i seek students i teach title i school students recei
ve free reduced price lunch despite disabilities limitations students love coming school
come eager learn explore have ever felt like ants pants needed groove move meeting this k
ids feel time the want able move learn say wobble chairs answer i love develop core enhan
ces gross motor turn fine motor skills they also want learn games kids not want sit works
heets they want learn count jumping playing physical engagement key success the number to
ss color shape mats make happen my students forget work fun 6 year old deserves nannan'
```

1.4 Preprocessing of `project_title`

Following Code blocks provided by me.

In [92]:

```
# Code took from original code provided.
# Also function used from original code.
preprocessed_titles = []

for sent in tqdm(project_data['project_title'].values):
    sent = decontracted(sent)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = ' '.join(e.lower() for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████| 109248/1092
48 [00:03<00:00, 33386.15it/s]
```

In [93]:

```
preprocessed_titles[20000]
```

Out[93]:

```
'need move input'
```

Following Code blocks present in original notebook.

1.5 Preparing data for models

In [94]:
```
project_data.columns
```

Out[94]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

```
    - school_state : categorical data
    - clean_categories : categorical data
    - clean_subcategories : categorical data
    - project_grade_category : categorical data
    - teacher_prefix : categorical data

    - project_title : text data
    - text : text data
    - project_resource_summary: text data (optinal)

    - quantity : numerical (optinal)
    - teacher_number_of_previously_posted_projects : numerical
    - price : numerical
```

1.5.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/ (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)

In [95]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, bina
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeed
s', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [96]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].value
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricu
lar', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hung
er', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'Co
llege_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopmen
t', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'Appli
edSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [97]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

Following Code blocks provided by me.

In [98]:

```
# Code took from original code provided.
states = project_data['school_state'].unique()
vectorizer = CountVectorizer(vocabulary=list(states), lowercase=False, binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())

school_state_one_hot = vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encoding", school_state_one_hot.shape)
```

```
['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY', 'OK', 'MA', 'NV', 'O
H', 'PA', 'AL', 'LA', 'VA', 'AR', 'WA', 'WV', 'ID', 'TN', 'MS', 'CO', 'UT', 'IL', 'MI',
'HI', 'IA', 'RI', 'NJ', 'MO', 'DE', 'MN', 'ME', 'WY', 'ND', 'OR', 'AK', 'MD', 'WI', 'SD',
'NE', 'NM', 'DC', 'KS', 'MT', 'NH', 'VT']
Shape of matrix after one hot encoding (109248, 51)
```

There are some NaN's in teacher_prefix column. replacing them with 'Mrs.' as that has high occurance in that column.

In [99]:

```
print("Number of NaN's before replacement in column: ", sum(project_data['teacher_prefix'].
project_data['teacher_prefix'] = project_data['teacher_prefix'].replace(np.nan, 'Mrs.', reg
print("Number of NaN's after replacement in column: ", sum(project_data['teacher_prefix'].i

# Output may show both zeros as I re-run this several times. But there are 3 zeros in origi
```

```
Number of NaN's before replacement in column:  3
Number of NaN's after replacement in column:  0
```

In [100]:

```python
# Code took from original code provided.
prefixes = project_data['teacher_prefix'].unique()
vectorizer = CountVectorizer(vocabulary=list(prefixes), lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())

teacher_prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].values)
print("Shape of matrix after one hot encoding", teacher_prefix_one_hot.shape)
```

```
['Mrs.', 'Mr.', 'Ms.', 'Teacher', 'Dr.']
Shape of matrix after one hot encoding (109248, 5)
```

In [101]:

```python
grades = project_data['project_grade_category'].unique()
vectorizer = CountVectorizer(vocabulary=list(grades), lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())

project_grade_category_one_hot = vectorizer.transform(project_data['project_grade_category'
print("Shape of matrix after one hot encoding", project_grade_category_one_hot.shape)
```

```
['Grades PreK-2', 'Grades 6-8', 'Grades 3-5', 'Grades 9-12']
Shape of matrix after one hot encoding (109248, 4)
```

Following Code blocks present in original notebook.

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [102]:

```
# We are considering only the words which appeared in at least 10 documents(rows or project
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

In [103]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

Following Code blocks provided by me.

In [104]:

```
# Code took from original code provided.
# We are considering only the words which appeared in at least 5 documents(rows or projects
# Reduced number as title has less words
vectorizer = CountVectorizer(min_df=10)
titles_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ", titles_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3222)
```

Following Code blocks present in original notebook.

1.5.2.2 TFIDF vectorizer

In [105]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [106]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickl
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [107]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████| 109248/109
248 [00:40<00:00, 2717.48it/s]

109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [108]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [109]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████| 109248/10
9248 [05:16<00:00, 344.76it/s]

109248
300
```

In [110]:

```
# Similarly you can vectorize for title also
```

Following Code blocks provided by me.

In [111]:

```
# Code took from original code provided.
# tfidf of project titles
vectorizer = TfidfVectorizer(min_df=10)
titles_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",titles_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3222)
```

In [112]:

```
# Code took from original code provided.
# avg-w2v for project titles
avg_w2v_titles = []
for sentence in tqdm(preprocessed_titles):
    vector = np.zeros(300)
    cnt_words =0;
    for word in sentence.split():
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles.append(vector)

print(len(avg_w2v_titles))
print(len(avg_w2v_titles[0]))
```

```
100%|████████████████████████████████████████████████| 109248/1092
48 [00:01<00:00, 57288.37it/s]

109248
300
```

```
In [113]: # Code took from original code provided.
          tfidf_model = TfidfVectorizer()
          tfidf_model.fit(preprocessed_titles)
          dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
          tfidf_words = set(tfidf_model.get_feature_names())
```

```
In [114]: # Code took from original code provided.
          # tfidf-w2v for project titles
          tfidf_w2v_titles = []
          for sentence in tqdm(preprocessed_titles):
              vector = np.zeros(300)
              tf_idf_weight =0
              for word in sentence.split():
                  if (word in glove_words) and (word in tfidf_words):
                      vec = model[word]
                      tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
                      vector += (vec * tf_idf)
                      tf_idf_weight += tf_idf
              if tf_idf_weight != 0:
                  vector /= tf_idf_weight
              tfidf_w2v_titles.append(vector)

          print(len(tfidf_w2v_titles))
          print(len(tfidf_w2v_titles[0]))
```

```
100%|████████████████████████████████████████████████████████████| 109248/1092
48 [00:04<00:00, 26231.39it/s]

109248
300
```

Following Code blocks present in original notebook.

1.5.3 Vectorizing Numerical features

In [115]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [116]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.prepro
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mean and standar
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

In [117]:
```
price_standardized
```

Out[117]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

Following Code blocks provided by me.

In [118]:

```
warnings.filterwarnings("ignore")
# Code took from original code provided
scalar = StandardScaler()
scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,
print(f"Mean : {scalar.mean_[0]}, Standard deviation : {np.sqrt(scalar.var_[0])}")

# Now standardize the data with above maen and variance.
previously_posted_projects_standardized = \
                scalar.transform(project_data['teacher_number_of_previously_posted_projects
print(previously_posted_projects_standardized)
```

```
Mean : 11.153165275336848, Standard deviation : 27.77702641477403
[[-0.40152481]
 [-0.14951799]
 [-0.36552384]
 ...
 [-0.29352189]
 [-0.40152481]
 [-0.40152481]]
```

Following Code blocks present in original notebook.

1.5.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [119]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [120]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[120]:

```
(109248, 16663)
```

In [121]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Computing Sentiment Scores

In [122]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest stu
for learning my students learn in many different ways using all of our senses and multiple
of techniques to help all my students succeed students in my class come from a variety of d
for wonderful sharing of experiences and cultures including native americans our school is
learners which can be seen through collaborative student project based learning in and out
in my class love to work with hands on materials and have many different opportunities to p
mastered having the social skills to work cooperatively with friends is a crucial aspect of
montana is the perfect place to learn about agriculture and nutrition my students love to r
in the early childhood classroom i have had several kids ask me can we try cooking with rea
and create common core cooking lessons where we learn important math and writing concepts w
food for snack time my students will have a grounded appreciation for the work that went in
of where the ingredients came from as well as how it is healthy for their bodies this proje
nutrition and agricultural cooking recipes by having us peel our own apples to make homemad
and mix up healthy plants from our classroom garden in the spring we will also create our o
shared with families students will gain math and literature skills as well as a life long e
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\narayana\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

Assignment 7: SVM

1. **[Task-1] Apply Support Vector Machines(SGDClassifier with hinge loss: Linear SVM) on these feature sets**

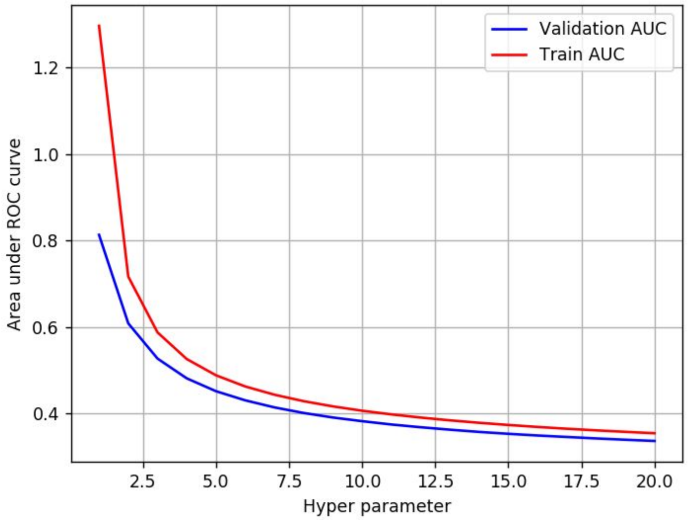
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

2. **The hyper paramter tuning (best alpha in range [10^-4 to 10^4], and the best penalty among 'l1', 'l2')**

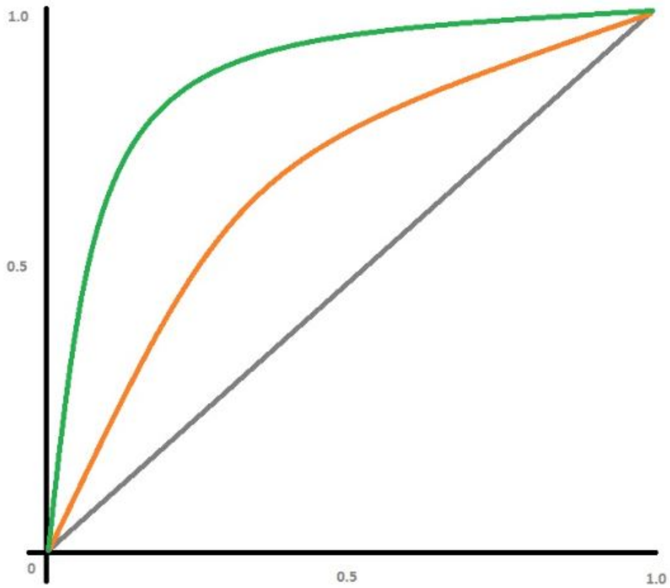
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data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

3. **Representation of results**

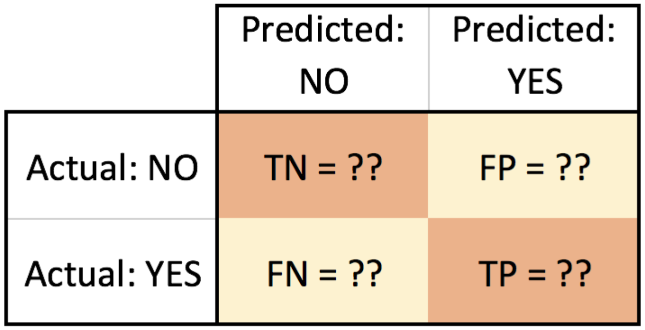
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure.



- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.



- Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.



	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

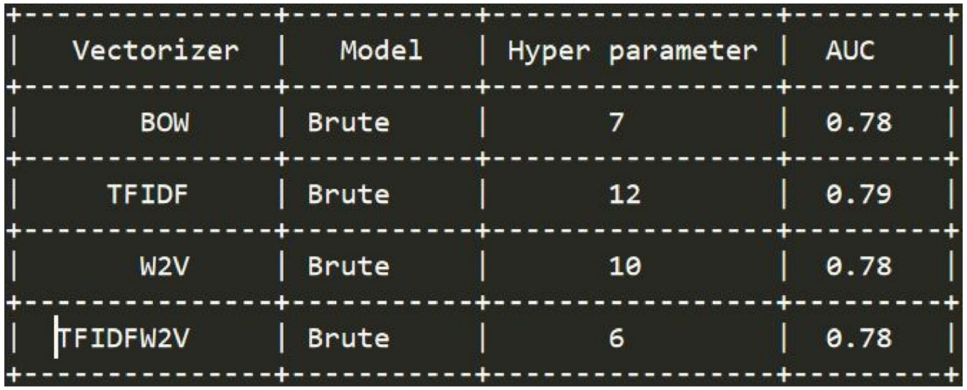
(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

4. **[Task-2] Apply the Support Vector Machines on these features by finding the best hyper paramter as suggested in step 2 and step 3**

- Consider these set of features Set 5 :
  - **school_state** : categorical data
  - **clean_categories** : categorical data
  - **clean_subcategories** : categorical data
  - **project_grade_category** :categorical data
  - **teacher_prefix** : categorical data
  - **quantity** : numerical data
  - **teacher_number_of_previously_posted_projects** : numerical data
  - **price** : numerical data
  - **sentiment score's of each of the essay** : numerical data
  - **number of words in the title** : numerical data
  - **number of words in the combine essays** : numerical data
  - **Apply TruncatedSVD (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html) on TfidfVectorizer (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) of essay text, choose the number of components (`n_components`) using elbow method**

(https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/pca-code-example-using-non-visualization/) : numerical data

- **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 (http://zetcode.com/python/prettytable/)



```
+--------------+-----------+-----------------+---------+
|  Vectorizer  |   Model   | Hyper parameter |  AUC    |
+--------------+-----------+-----------------+---------+
|     BOW      | Brute     |        7        |  0.78   |
+--------------+-----------+-----------------+---------+
|    TFIDF     | Brute     |        12       |  0.79   |
+--------------+-----------+-----------------+---------+
|     W2V      | Brute     |        10       |  0.78   |
+--------------+-----------+-----------------+---------+
|   TFIDFW2V   | Brute     |        6        |  0.78   |
+--------------+-----------+-----------------+---------+
```

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-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. Support Vector Machines

Some code blocks are taken from previous assignments. And some used the code present in original file ('7_DonorsChoose_SVM.ipynb') which is mentioned in comments.

Following Code blocks provided by me.

Adding a column `summary_numeric_bool` instead of `project_resource_summary` column which tells if resource summary has a number in it

In [123]:

```python
# ref: https://stackoverflow.com/questions/4138202/using-isdigit-for-floats
def nums_in_str(text):
    """
    Returns list of numbers present in the given string. Numbers := floats ints etc.
    """
    result = []
    for s in text.split():
        try:
            x = float(s)
            result.append(x)
        except:
            continue
    return result
```

In [124]:

```python
print(nums_in_str('HE44LLo 56 are -89 I 820.353 in -78.39 what .293 about 00'))
```

```
[56.0, -89.0, 820.353, -78.39, 0.293, 0.0]
```

In [125]:

```python
numbers_in_summary = np.array([len(nums_in_str(s)) for s in project_data['project_resource_
project_data['summary_numeric_bool'] = list(map(int, numbers_in_summary>0))
```

Taking Relevant columns as X (input data to model) and y (output class

label)

In [126]:
```
project_data.columns
```

Out[126]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'summary_numeric_bool'],
      dtype='object')
```

In [127]:
```
project_data.head(2)
```

Out[127]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datet
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:2

2 rows × 21 columns

In [128]:

```
# Categorical and numerical columns are listed below.
X_columns = ['teacher_prefix', 'school_state', 'project_grade_category', 'summary_numeric_b
             'teacher_number_of_previously_posted_projects', 'clean_categories', 'clean_sub
             'price', 'quantity']
X = project_data[X_columns]
y = project_data['project_is_approved']
```

Adding preprocessed_essays and preprocessed_titles as columns to X before splitting

In [129]:

```
X['essay'] = preprocessed_essays
X['project_title'] = preprocessed_titles
X_columns.append('essay')
X_columns.append('project_title')
print('final columns used in input data are: ', X_columns)
```

```
final columns used in input data are:  ['teacher_prefix', 'school_state', 'project_grade_
category', 'summary_numeric_bool', 'teacher_number_of_previously_posted_projects', 'clean
_categories', 'clean_subcategories', 'price', 'quantity', 'essay', 'project_title']
```

Adding essays and calculating sentiments to Input data X before splitting as we have to use same train and test rows later for Task-2 analysis. These columns are not considered in our Task-1 analysis

In [130]:

```
X['essay_1'] = project_data['project_essay_1']
X['essay_2'] = project_data['project_essay_2']
X['essay_3'] = project_data['project_essay_3']
X['essay_4'] = project_data['project_essay_4']
```

In [131]:

```
sia = SentimentIntensityAnalyzer()
for esnum in range(1, 5):
    sentim_data = []
    for es in project_data['project_essay_' + str(esnum)]:
        sentim_data.append(list(sia.polarity_scores(str(es)).values()))
    df_cols = ['essay' + str(esnum) + '_neg', 'essay' + str(esnum) + '_nue',\
               'essay' + str(esnum) + '_pos', 'essay' + str(esnum) + '_comp']
    sentim_data = pd.DataFrame(sentim_data, columns=df_cols)
    X = pd.concat([X, sentim_data], axis=1)
```

In [132]:

```
X['essay_word_count'] = [len(es.split()) for es in X['essay']]
X['title_word_count'] = [len(title.split()) for title in X['project_title']]
```

In [133]:

```
print(X.columns)
```

```
Index(['teacher_prefix', 'school_state', 'project_grade_category',
       'summary_numeric_bool', 'teacher_number_of_previously_posted_projects',
       'clean_categories', 'clean_subcategories', 'price', 'quantity', 'essay',
       'project_title', 'essay_1', 'essay_2', 'essay_3', 'essay_4',
       'essay1_neg', 'essay1_nue', 'essay1_pos', 'essay1_comp', 'essay2_neg',
       'essay2_nue', 'essay2_pos', 'essay2_comp', 'essay3_neg', 'essay3_nue',
       'essay3_pos', 'essay3_comp', 'essay4_neg', 'essay4_nue', 'essay4_pos',
       'essay4_comp', 'essay_word_count', 'title_word_count'],
      dtype='object')
```

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [134]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Not creating CV data as I am using K-fold validation

In [135]:

```
# Code took from SAMPLE_SOLUTION notebook
# splitting into 80-20 ratio for train-test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, stratify=y)
```

In [136]:

```
print(X_train.shape)
print(X_test.shape)
print('='*30)
print(y_train.shape)
print(y_test.shape)
```

```
(87398, 33)
(21850, 33)
==============================
(87398,)
(21850,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [137]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

numerical columns

- `teacher_number_of_previously_posted_projects`
- `price`
- `quantity`

Leaving `summary_numeric_bool` as it is because it only has 0's and 1's in it.

categorical columns

- `teacher_prefix`
- `school_state`
- `project_grade_category`
- `clean_categories`
- `clean_subcategories`

Normalizing `teacher_number_of_previously_posted_projects` column

In [138]:

```
warnings.filterwarnings("ignore")
# Code took from original Code provided.
scaler = StandardScaler()
scaler.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
print(f"Mean : {scaler.mean_[0]}, Standard deviation : {np.sqrt(scaler.var_[0])}")
```

```
Mean : 11.169019886038582, Standard deviation : 27.94816864975853
```

In [139]:

```
warnings.filterwarnings("ignore")
X_train_tnppp_norm = scaler.transform(X_train['teacher_number_of_previously_posted_projects
X_test_tnppp_norm = scaler.transform(X_test['teacher_number_of_previously_posted_projects']
```

Normalizing price column

In [140]:

```
# Code took from original Code provided.
scaler = StandardScaler()
scaler.fit(X_train['price'].values.reshape(-1,1))
print(f"Mean : {scaler.mean_[0]}, Standard deviation : {np.sqrt(scaler.var_[0])}")
```

```
Mean : 297.29548948488525, Standard deviation : 368.33513488297683
```

In [141]:

```
X_train_price_norm = scaler.transform(X_train['price'].values.reshape(-1,1))
X_test_price_norm = scaler.transform(X_test['price'].values.reshape(-1,1))
```

Normalizing quantity column

In [142]:

```
warnings.filterwarnings("ignore")
# Code took from original Code provided.
scaler = StandardScaler()
scaler.fit(X_train['quantity'].values.reshape(-1,1))
print(f"Mean : {scaler.mean_[0]}, Standard deviation : {np.sqrt(scaler.var_[0])}")
```

```
Mean : 16.95863749742557, Standard deviation : 26.370536980731185
```

In [143]:

```
warnings.filterwarnings("ignore")
X_train_quant_norm = scaler.transform(X_train['quantity'].values.reshape(-1,1))
X_test_quant_norm = scaler.transform(X_test['quantity'].values.reshape(-1,1))
```

Encoding teacher_prefix column

In [144]:

```
# Code took from SAMPLE_SOLUTION notebook.
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values)
print(vectorizer.get_feature_names())
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
```

In [145]:

```
# Code took from SAMPLE_SOLUTION notebook.
X_train_prefix_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_test_prefix_ohe = vectorizer.transform(X_test['teacher_prefix'].values)

print(X_train_prefix_ohe.shape, y_train.shape)
print(X_test_prefix_ohe.shape, y_test.shape)
```

```
(87398, 5) (87398,)
(21850, 5) (21850,)
```

Encoding school_state column

In [146]:

```
# Code took from SAMPLE_SOLUTION notebook.
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values)
print(vectorizer.get_feature_names())
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'i
l', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd',
'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx',
'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
```

In [147]:

```
# Code took from SAMPLE_SOLUTION notebook.
X_train_school_ohe = vectorizer.transform(X_train['school_state'].values)
X_test_school_ohe = vectorizer.transform(X_test['school_state'].values)

print(X_train_school_ohe.shape, y_train.shape)
print(X_test_school_ohe.shape, y_test.shape)
```

```
(87398, 51) (87398,)
(21850, 51) (21850,)
```

Encoding project_grade_category column

In [148]:

```
# Code took from original Code provided.
grades = X_train['project_grade_category'].unique()
vectorizer = CountVectorizer(vocabulary=list(grades), lowercase=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values)
print(vectorizer.get_feature_names())
```

```
['Grades 9-12', 'Grades 3-5', 'Grades PreK-2', 'Grades 6-8']
```

In [149]:

```
# Code took from SAMPLE_SOLUTION notebook.
X_train_grade_ohe = vectorizer.transform(X_train['project_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade_category'].values)

print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
```

```
(87398, 4) (87398,)
(21850, 4) (21850,)
```

Encoding clean_categories column

In [150]:

```
# Code took from original Code provided.
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, bina
vectorizer.fit(X_train['clean_categories'].values)
print(vectorizer.get_feature_names())
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeed
s', 'Health_Sports', 'Math_Science', 'Literacy_Language']
```

In [151]:

```
# Code took from SAMPLE_SOLUTION notebook.
X_train_categ_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_test_categ_ohe = vectorizer.transform(X_test['clean_categories'].values)

print(X_train_categ_ohe.shape, y_train.shape)
print(X_test_categ_ohe.shape, y_test.shape)
```

```
(87398, 9) (87398,)
(21850, 9) (21850,)
```

Encoding clean_subcategories column

In [152]:

```
# Code took from original Code provided.
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
vectorizer.fit(X_train['clean_subcategories'].values)
print(vectorizer.get_feature_names())
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricu
lar', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hung
er', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'Co
llege_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopmen
t', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'Appli
edSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
```

In [153]:

```
# Code took from SAMPLE_SOLUTION notebook.
X_train_subcat_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_test_subcat_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print(X_train_subcat_ohe.shape, y_train.shape)
print(X_test_subcat_ohe.shape, y_test.shape)
```

```
(87398, 30) (87398,)
(21850, 30) (21850,)
```

Combining categorical and numerical data for further use.

In [154]:

```
from scipy.sparse import hstack
cat_num_train = hstack((X_train_tnppp_norm, X_train_price_norm, X_train_quant_norm,\
                        np.array(X_train['summary_numeric_bool']).reshape(-1, 1),\
                        X_train_prefix_ohe, X_train_grade_ohe, X_train_school_ohe, X_train_
cat_num_test = hstack((X_test_tnppp_norm, X_test_price_norm, X_test_quant_norm,\
                       np.array(X_test['summary_numeric_bool']).reshape(-1, 1),\
                       X_test_prefix_ohe, X_test_grade_ohe, X_test_school_ohe, X_test_categ
```

In [155]:

```
print(cat_num_train.shape, y_train.shape)
print(cat_num_test.shape, y_test.shape)
```

```
(87398, 103) (87398,)
(21850, 103) (21850,)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [156]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Converting essay column to vector using Bag of Words (BoW).

In [157]:

```
# Code took from original Code provided.
vectorizer = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
vectorizer.fit(X_train['essay'].values)
print(len(vectorizer.get_feature_names()))
```

```
5000
```

In [158]:

```python
# Code took from SAMPLE_SOLUTION notebook.
X_train_essay_bow = vectorizer.transform(X_train['essay'].values)
X_test_essay_bow = vectorizer.transform(X_test['essay'].values)

print(X_train_essay_bow.shape, y_train.shape)
print(X_test_essay_bow.shape, y_test.shape)
```

```
(87398, 5000) (87398,)
(21850, 5000) (21850,)
```

Converting `essay` column to vector using TFIDF Vectorizer.

In [159]:

```python
# Code took from original Code provided.
vectorizer = TfidfVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
vectorizer.fit(X_train['essay'].values)
print(len(vectorizer.get_feature_names()))
```

```
5000
```

In [160]:

```python
# Code took from SAMPLE_SOLUTION notebook.
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].values)

print(X_train_essay_tfidf.shape, y_train.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
```

```
(87398, 5000) (87398,)
(21850, 5000) (21850,)
```

Converting `essay` column to vector using Average Word2Vec.

Creating function to return average word2vec vectors given sentences

In [161]:

```
# Code took from original Code provided.
def avg_w2v(arr):
    """
    Returns array of vectors given array of sentences. Array of vectors are created by Aver
    words is taken from 'glove_vectors' file.
    """
    avg_w2v_vectors = []
    for sentence in tqdm(arr):
        vector = np.zeros(300)
        cnt_words = 0
        for word in sentence.split():
            if word in glove_words:
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_vectors.append(vector)
    return avg_w2v_vectors
```

In [162]:

```
X_train_essay_avgw2v = np.array(avg_w2v(X_train['essay'].values))
X_test_essay_avgw2v = np.array(avg_w2v(X_test['essay'].values))

print(X_train_essay_avgw2v.shape, y_train.shape)
print(X_test_essay_avgw2v.shape, y_test.shape)
```

```
100%|██████████████████████████████████████| 87398/87
398 [00:32<00:00, 2672.31it/s]
100%|██████████████████████████████████████| 21850/21
850 [00:07<00:00, 2817.94it/s]

(87398, 300) (87398,)
(21850, 300) (21850,)
```

Converting essay column to vector using TFIDF weighted Word2Vec.

Creating function to return tfidf weighted word2vec vectors given sentences and idf dictionary for words

In [163]:

```
# Code took from original Code provided.
def tfidf_w2v(arr, idf_dict):
    """
    Returns array of vectors given array of sentences and dictionary containing IDF values
    Array of vectors are created by TFIDF weighted Word2Vec method and vectors for words is
    """
    tfidf_w2v_vectors = []
    for sentence in tqdm(arr):
        vector = np.zeros(300)
        tf_idf_weight = 0;
        for word in sentence.split():
            if (word in glove_words) and (word in idf_dict):
                vec = model[word]
                tf_idf = idf_dict[word]/len(sentence.split())
                vector += (vec * tf_idf)
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors.append(vector)
    return tfidf_w2v_vectors
```

Getting idf values for the words in X_train.essay data

In [164]:
```
# Code took from original Code provided.
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
```

In [165]:
```
X_train_essay_tfidfw2v = np.array(tfidf_w2v(X_train['essay'].values, dictionary))
X_test_essay_tfidfw2v = np.array(tfidf_w2v(X_test['essay'].values, dictionary))

print(X_train_essay_tfidfw2v.shape, y_train.shape)
print(X_test_essay_tfidfw2v.shape, y_test.shape)
```

```
100%|██████████████████████████████████████████████████| 87398/8
7398 [03:28<00:00, 418.99it/s]
100%|██████████████████████████████████████████████████| 21850/2
1850 [00:52<00:00, 416.73it/s]

(87398, 300) (87398,)
(21850, 300) (21850,)
```

Converting `project_title` column to vector using Bag of Words (BoW).

In [166]:
```
# Code took from original Code provided.
vectorizer = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
vectorizer.fit(X_train['project_title'].values)
print(len(vectorizer.get_feature_names()))
```

```
5000
```

In [167]:

```
# Code took from SAMPLE_SOLUTION notebook.
X_train_title_bow = vectorizer.transform(X_train['project_title'].values)
X_test_title_bow = vectorizer.transform(X_test['project_title'].values)

print(X_train_title_bow.shape, y_train.shape)
print(X_test_title_bow.shape, y_test.shape)
```

```
(87398, 5000) (87398,)
(21850, 5000) (21850,)
```

Converting `project_title` column to vector using TFIDF Vectorizer.

In [168]:

```
# Code took from original Code provided.
vectorizer = TfidfVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
vectorizer.fit(X_train['project_title'].values)
print(len(vectorizer.get_feature_names()))
```

```
5000
```

In [169]:

```
# Code took from SAMPLE_SOLUTION notebook.
X_train_title_tfidf = vectorizer.transform(X_train['project_title'].values)
X_test_title_tfidf = vectorizer.transform(X_test['project_title'].values)

print(X_train_title_tfidf.shape, y_train.shape)
print(X_test_title_tfidf.shape, y_test.shape)
```

```
(87398, 5000) (87398,)
(21850, 5000) (21850,)
```

Converting `project_title` column to vector using Average Word2Vec.

Can use avg_w2v function

In [170]:

```
X_train_title_avgw2v = np.array(avg_w2v(X_train['project_title'].values))
X_test_title_avgw2v = np.array(avg_w2v(X_test['project_title'].values))

print(X_train_title_avgw2v.shape, y_train.shape)
print(X_test_title_avgw2v.shape, y_test.shape)
```

```
100%|█████████████████████████████████████████████████████████████████████| 87398/873
98 [00:01<00:00, 54380.94it/s]
100%|█████████████████████████████████████████████████████████████████████| 21850/218
50 [00:00<00:00, 63180.42it/s]

(87398, 300) (87398,)
(21850, 300) (21850,)
```

Converting `project_title` column to vector using TFIDF weighted Word2Vec.

Can use tfidf_w2v function but should calculate idf dictionary before using it

In [171]:

```
# Code took from original Code provided.
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
```

In [172]:

```
X_train_title_tfidfw2v = np.array(tfidf_w2v(X_train['project_title'].values, dictionary))
X_test_title_tfidfw2v = np.array(tfidf_w2v(X_test['project_title'].values, dictionary))

print(X_train_title_tfidfw2v.shape, y_train.shape)
print(X_test_title_tfidfw2v.shape, y_test.shape)
```

```
100%|████████████████████████████████████████| 87398/873
98 [00:03<00:00, 28698.56it/s]
100%|████████████████████████████████████████| 21850/218
50 [00:00<00:00, 26563.36it/s]

(87398, 300) (87398,)
(21850, 300) (21850,)
```

2.4 Appling Support Vector Machines on different kind of featurization as mentioned in the instructions

Apply Support Vector Machines on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

In [173]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [174]:

```
bow_train = hstack((cat_num_train, X_train_essay_bow, X_train_title_bow)).tocsr()
bow_test = hstack((cat_num_test, X_test_essay_bow, X_test_title_bow)).tocsr()

tfidf_train = hstack((cat_num_train, X_train_essay_tfidf, X_train_title_tfidf)).tocsr()
tfidf_test = hstack((cat_num_test, X_test_essay_tfidf, X_test_title_tfidf)).tocsr()

avgw2v_train = np.hstack((cat_num_train.toarray(), X_train_essay_avgw2v, X_train_title_avgw
avgw2v_test = np.hstack((cat_num_test.toarray(), X_test_essay_avgw2v, X_test_title_avgw2v))

tfidfw2v_train = np.hstack((cat_num_train.toarray(), X_train_essay_tfidfw2v, X_train_title_
tfidfw2v_test = np.hstack((cat_num_test.toarray(), X_test_essay_tfidfw2v, X_test_title_tfid

print('='*30)
print(bow_train.shape)
print(bow_test.shape)
print('='*30)
print(tfidf_train.shape)
print(tfidf_test.shape)
print('='*30)
print(avgw2v_train.shape)
print(avgw2v_test.shape)
print('='*30)
print(tfidfw2v_train.shape)
print(tfidfw2v_test.shape)
print('='*30)
```

```
==============================
(87398, 10103)
(21850, 10103)
==============================
(87398, 10103)
(21850, 10103)
==============================
(87398, 703)
(21850, 703)
```

```
==============================
(87398, 703)
(21850, 703)
==============================
```

Writing several functions to reuse them later

Function to plot AUC values with respect to hyper-parameter C given train data using K-fold validation

In [175]:

```
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
import math

# Code inside function took from SAMPLE_SOLUTION notebook
def auc_vs_K_plot(X_train, y_train, alphas, penalty='l2', logplot=True):
    """
    Plots the AUC results for different alpha values on train and CV data
    Parameters:
    X_train, y_train - data which is used for K-fold validation and used to train SGDClassi
    alphas - list of alpha values on which we have to train the data and plot the results
    penalty - which regularization to use
    """
    svm_model = SGDClassifier(loss='hinge', penalty=penalty)
    parameters = {'alpha': alphas}
    clf = GridSearchCV(svm_model, parameters, cv=3, scoring='roc_auc')
    clf.fit(X_train, y_train)

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

    plt.figure(figsize=(12, 6))
    if logplot:
        # taking logs of alphas to plot a log-plot
        x_axis_ticks = [math.log10(i) for i in alphas]
    else:
        x_axis_ticks = alphas
    plt.plot(x_axis_ticks, train_auc, label='Train AUC')
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(x_axis_ticks, train_auc - train_auc_std, train_auc + train_auc_s

    plt.plot(x_axis_ticks, cv_auc, label='CV AUC')
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(x_axis_ticks, cv_auc - cv_auc_std, cv_auc + cv_auc_std, alpha=0.

    plt.scatter(x_axis_ticks, train_auc, label='Train AUC points')
    plt.scatter(x_axis_ticks, cv_auc, label='CV AUC points')

    plt.legend()
    plt.xlabel("alpha")
    # Setting x-ticks to match with actual alpha calues
    if logplot:
        plt.xticks(x_axis_ticks, ["{:.0e}".format(i) for i in alphas])
    plt.ylabel("AUC")
    plt.title("AUC PLOTS for train and CV data")
    plt.grid()
    plt.show()
```

Function to plots ROC curves and confusion matrices for train and test data. Function returns AUC Values for train, test data

In Function using CalibratedClassifierCV to predict probability of SGDClassifier with hinge loss

In [183]:

```
from sklearn.metrics import roc_curve, auc, precision_recall_curve
from sklearn.calibration import CalibratedClassifierCV
from IPython.display import Markdown, display

# Code inside function took from SAMPLE_SOLUTION notebook
def ROC_conf_mat(X_train, y_train, X_test, y_test, best_alpha, penalty='l2', plots = True):
    """
    Plots ROC Curve given a C value, Train data and Test data using LogisticRegression.
    And also plots confusion matrix for train data and test data taking a optimal threshold
    Returns Area Under ROC Curve for Train, Test data which can be taken as performance of
    """
    # Plotting ROC Curve code
    svm_model = SGDClassifier(loss='hinge', penalty=penalty, alpha = best_alpha)
    svm_model = CalibratedClassifierCV(svm_model)
    svm_model.fit(X_train, y_train)

    y_train_pred = svm_model.predict_proba(X_train)[:, 1]
    y_test_pred = svm_model.predict_proba(X_test)[:, 1]

    train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
    test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

    result = {}

    result['train_auc'], result['test_auc'] = (auc(train_fpr, train_tpr), auc(test_fpr, tes

    if(plots):
        display(Markdown(f"**Analysis for alpha = {best_alpha}**"))

        plt.plot(train_fpr, train_tpr, label="train AUC ="+str(np.round(result['train_auc']
        plt.plot(test_fpr, test_tpr, label="test AUC ="+str(np.round(result['test_auc'], 3)
        plt.legend()
        plt.xlabel("False Positive rate")
        plt.ylabel("True Positive rate")
        plt.title("ROC Curves for Train and Test data")
        plt.grid()
        plt.show()

        # Printing confusion matrices code
        thr_train = tr_thresholds[np.argmax(train_tpr*(1-train_fpr))]
        thr_test = te_thresholds[np.argmax(test_tpr*(1-test_fpr))]

        print(f"\nConfusion matrix for Train data with {thr_train} as threshold:")
        predictions = []
        for i in y_train_pred:
            if i >= thr_train:
                predictions.append(1)
            else:
                predictions.append(0)
        ax = sns.heatmap(confusion_matrix(y_train, predictions), annot=True, fmt='g')
        ax.set_yticklabels(['Rejected', 'Accepted'])
        ax.set_xticklabels(['Rejected', 'Accepted'])
        plt.xlabel("Predicted")
        plt.ylabel("Actual")
        plt.title('Confusion matrix for Train')
        plt.show()

        print(f"\nConfusion matrix for Test data with {thr_test} as threshold:")
        predictions = []
        for i in y_test_pred:
            if i >= thr_test:
                predictions.append(1)
            else:
                predictions.append(0)
        ax = sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt='g')
        ax.set_yticklabels(['Rejected', 'Accepted'])
        ax.set_xticklabels(['Rejected', 'Accepted'])
        plt.xlabel("Predicted")
        plt.ylabel("Actual")
        plt.title('Confusion matrix for Test')
        plt.show()
```

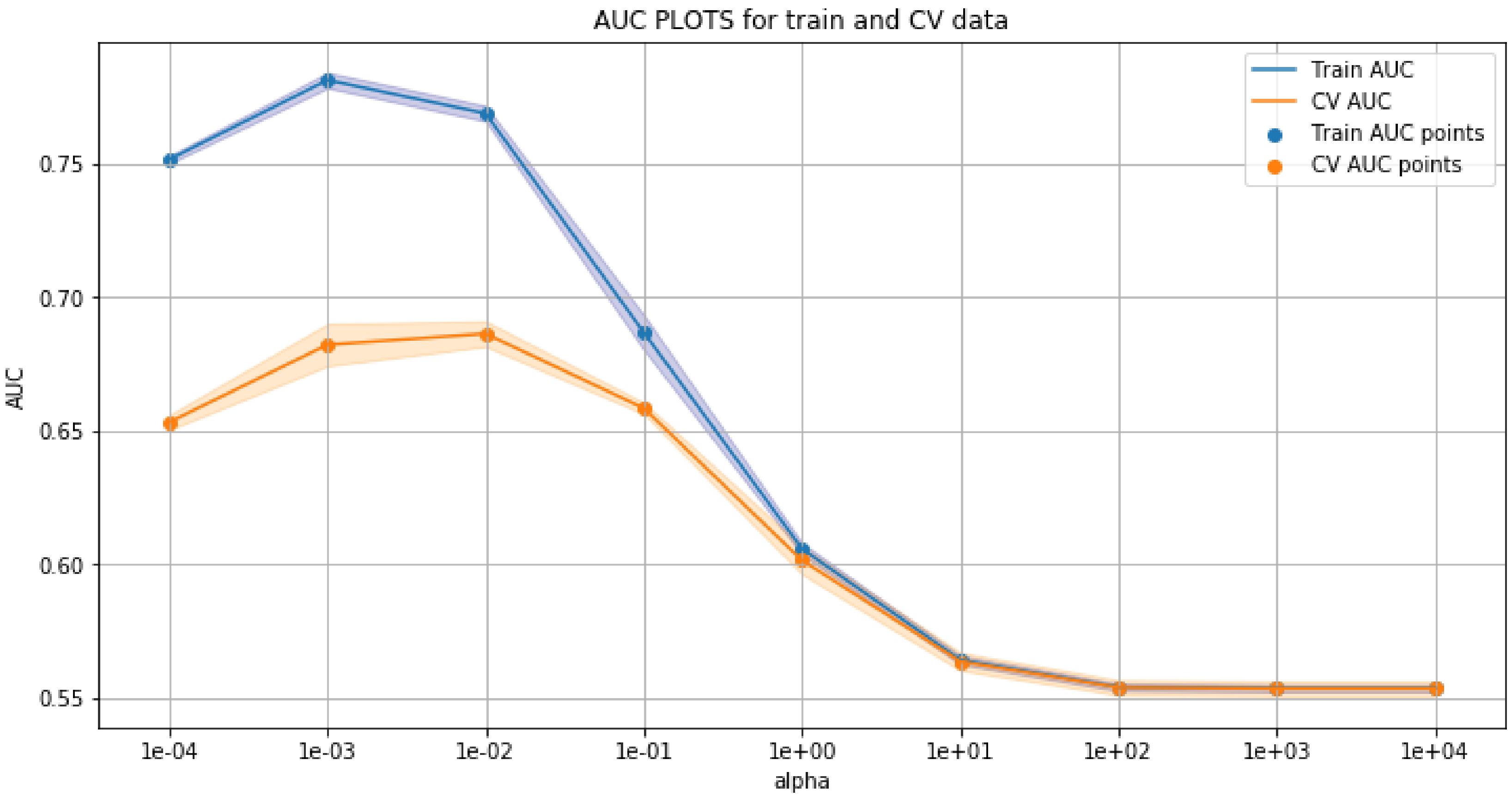
```
    return result
```

2.4.1 Applying SVM on BOW, SET 1

With L2 Penalty

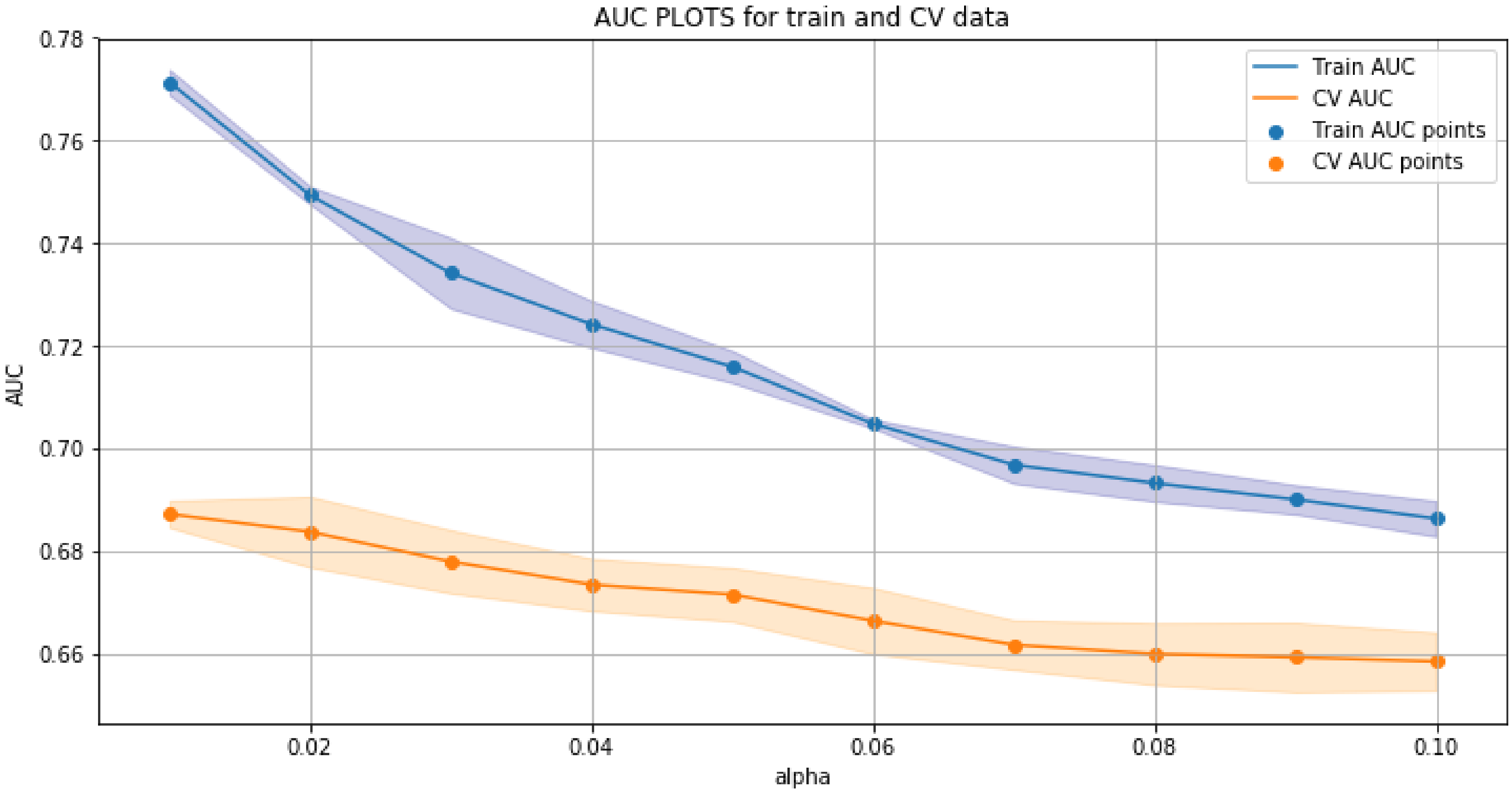
In [179]:

```
alphas = [10**i for i in range(-4, 5)]
auc_vs_K_plot(bow_train, y_train, alphas, penalty='l2', logplot=True)
```



Taking alpha range from [0.01, 0.1] to find best alpha

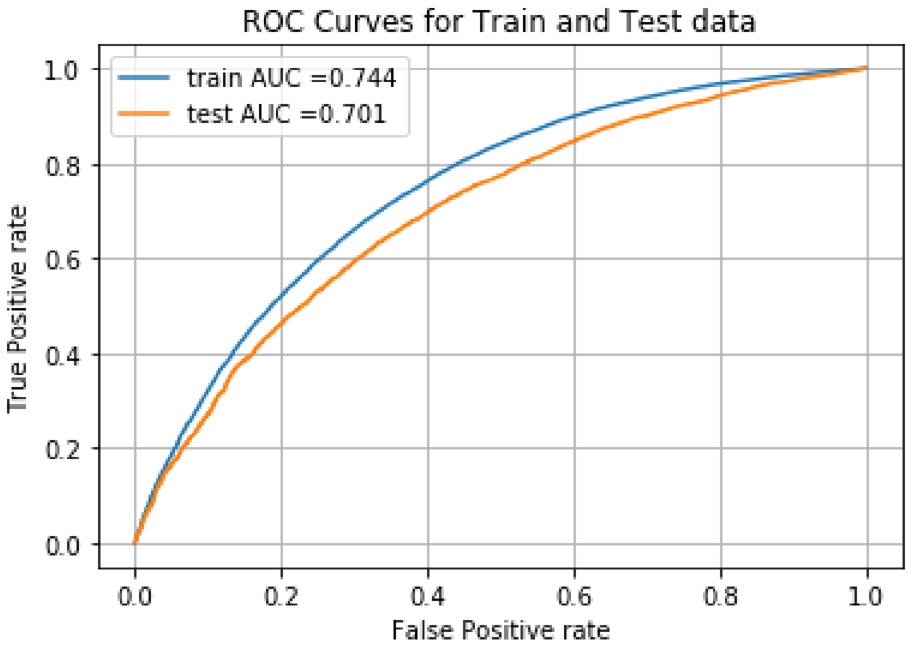
```
In [180]: alphas = np.arange(0.01, 0.11, 0.01)
          auc_vs_K_plot(bow_train, y_train, alphas, penalty='l2', logplot=False)
```



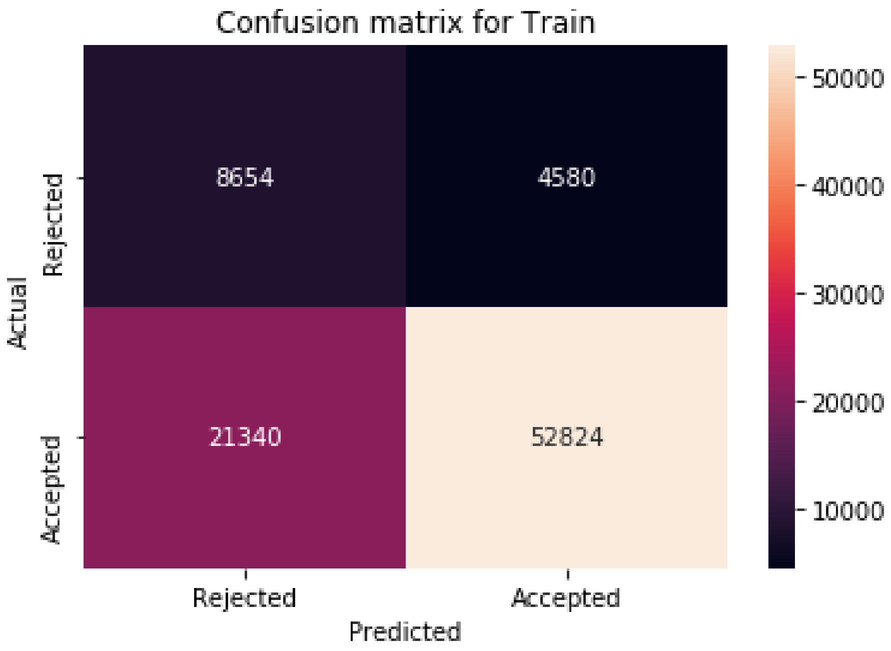
Taking best alpha = 0.02

```
In [184]: bow_l2_result = {}
In [185]: bow_l2_result[0.02] = ROC_conf_mat(bow_train, y_train, bow_test, y_test, 0.02, penalty='l2'
```

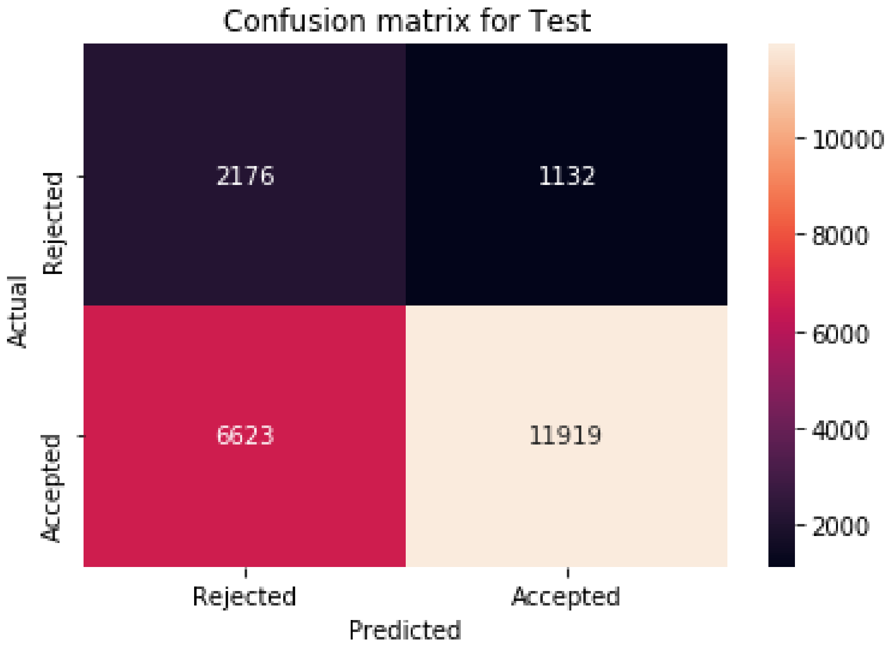
Analysis for alpha = 0.02



```
Confusion matrix for Train data with 0.8282850599806476 as threshold:
```

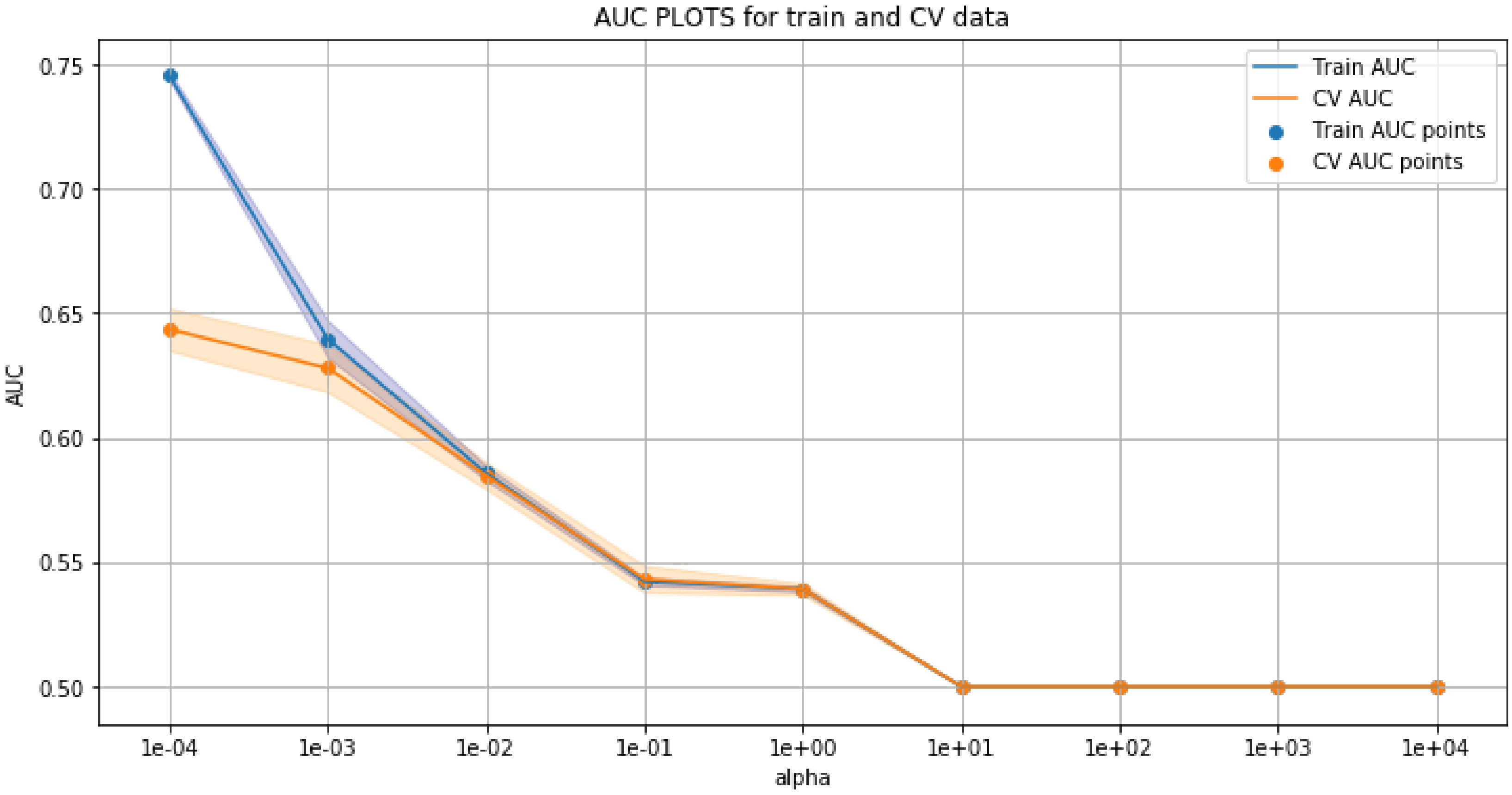


```
Confusion matrix for Test data with 0.8411502035860945 as threshold:
```



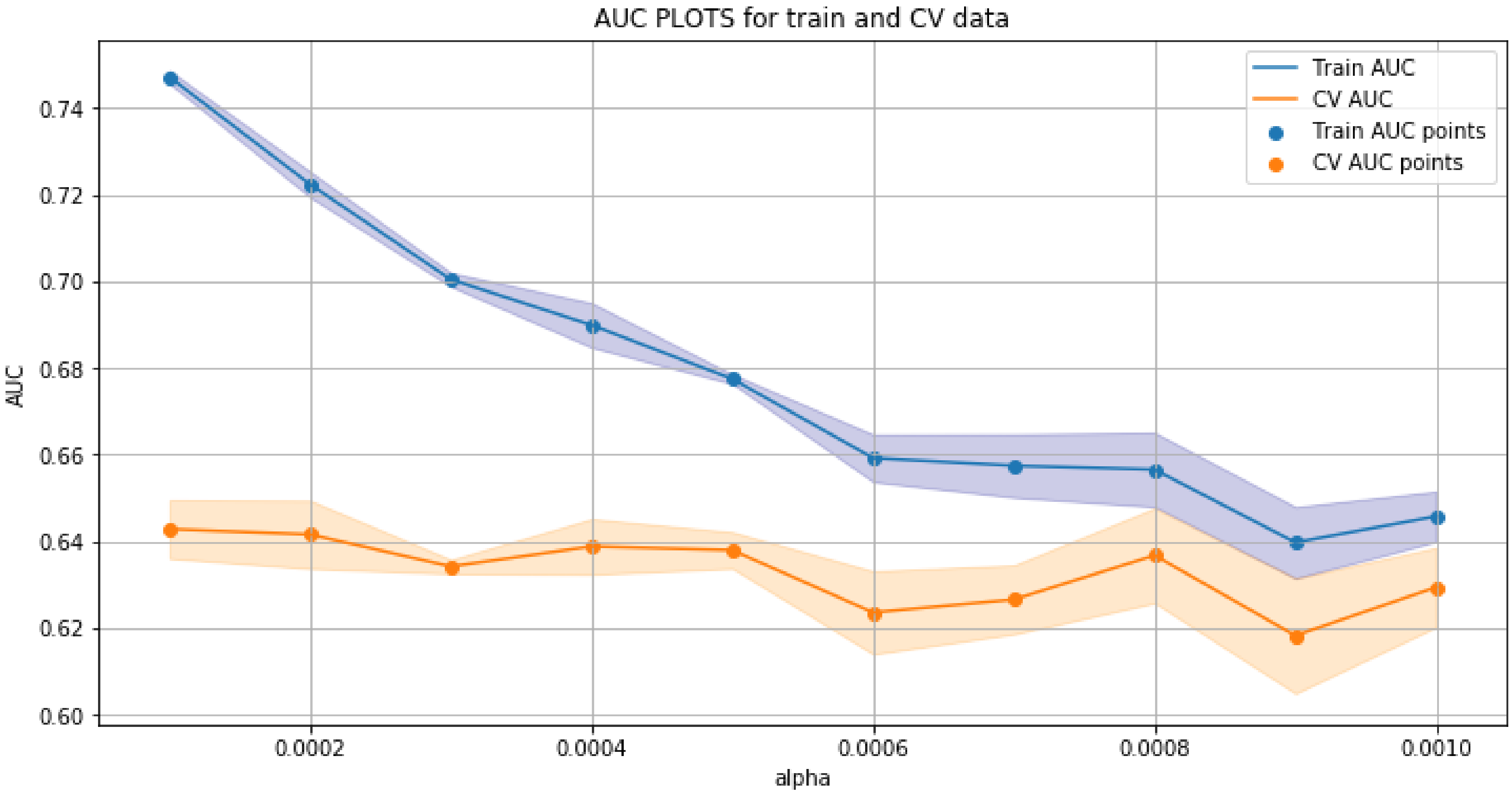
With L1 Penalty

In [186]:
```
alphas = [10**i for i in range(-4, 5)]
auc_vs_K_plot(bow_train, y_train, alphas, penalty='l1', logplot=True)
```



Taking Range [0.0001, 0.001]

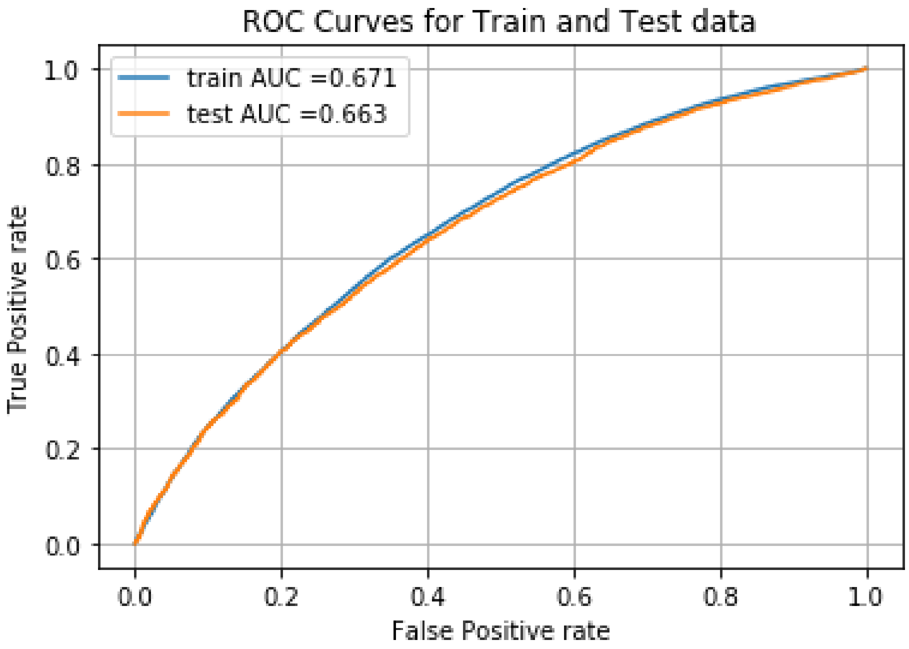
In [189]:
```
alphas = np.arange(0.0001, 0.0011, 0.0001)
auc_vs_K_plot(bow_train, y_train, alphas, penalty='l1', logplot=False)
```



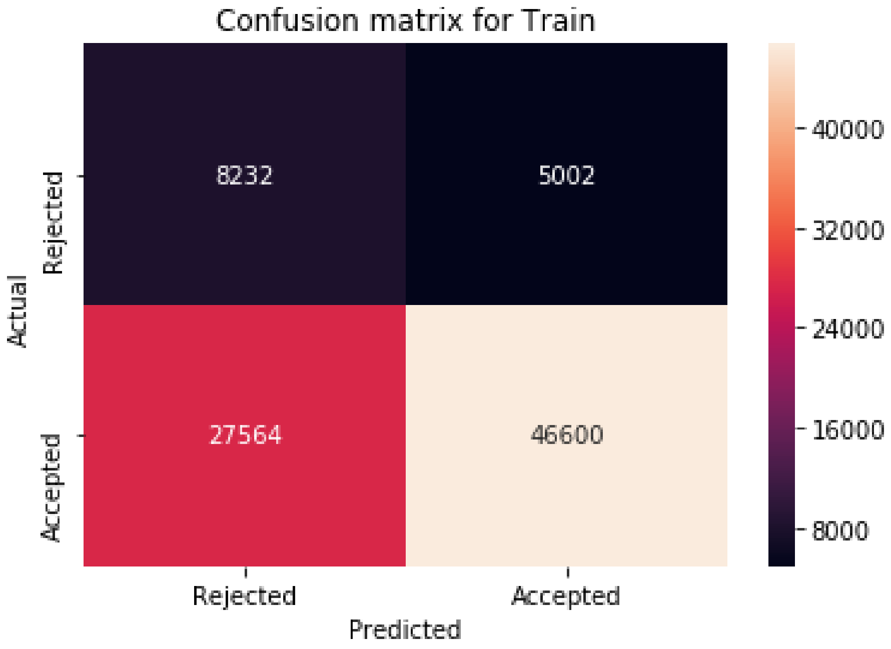
best alpha = 0.0008

In [190]:
```
bow_l1_result = {}
bow_l1_result[0.0008] = ROC_conf_mat(bow_train, y_train, bow_test, y_test, 0.0008, penalty=
```

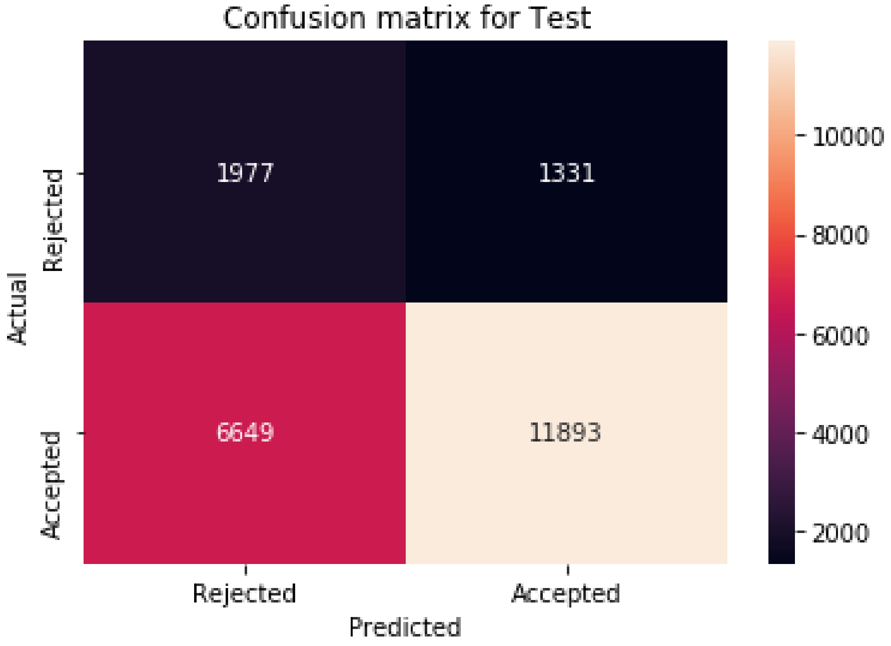
Analysis for alpha = 0.0008



```
Confusion matrix for Train data with 0.8406632653678346 as threshold:
```



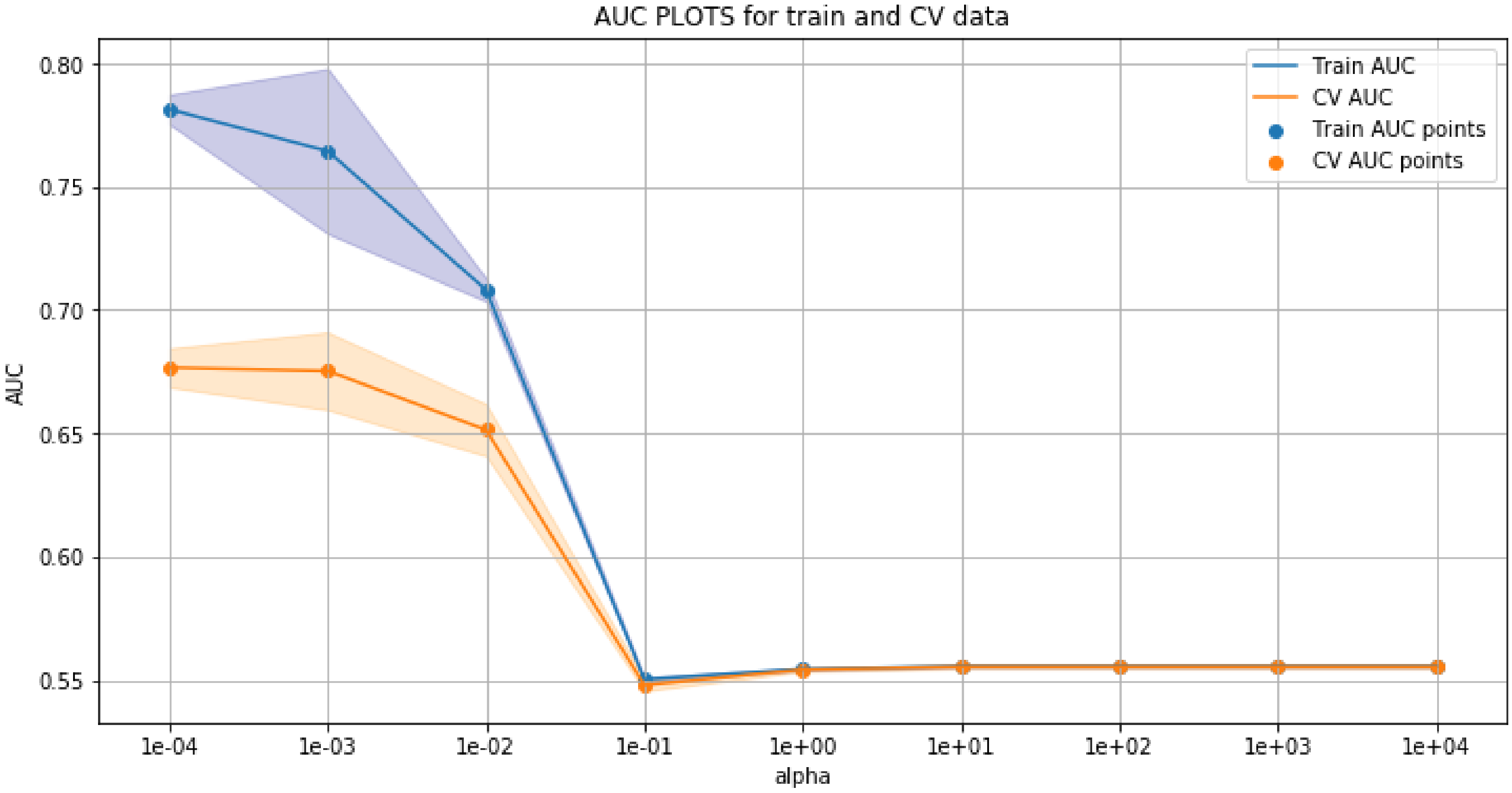
```
Confusion matrix for Test data with 0.8398124357986387 as threshold:
```



2.4.2 Applying SVM on TFIDF, SET 2

With L2 Penalty

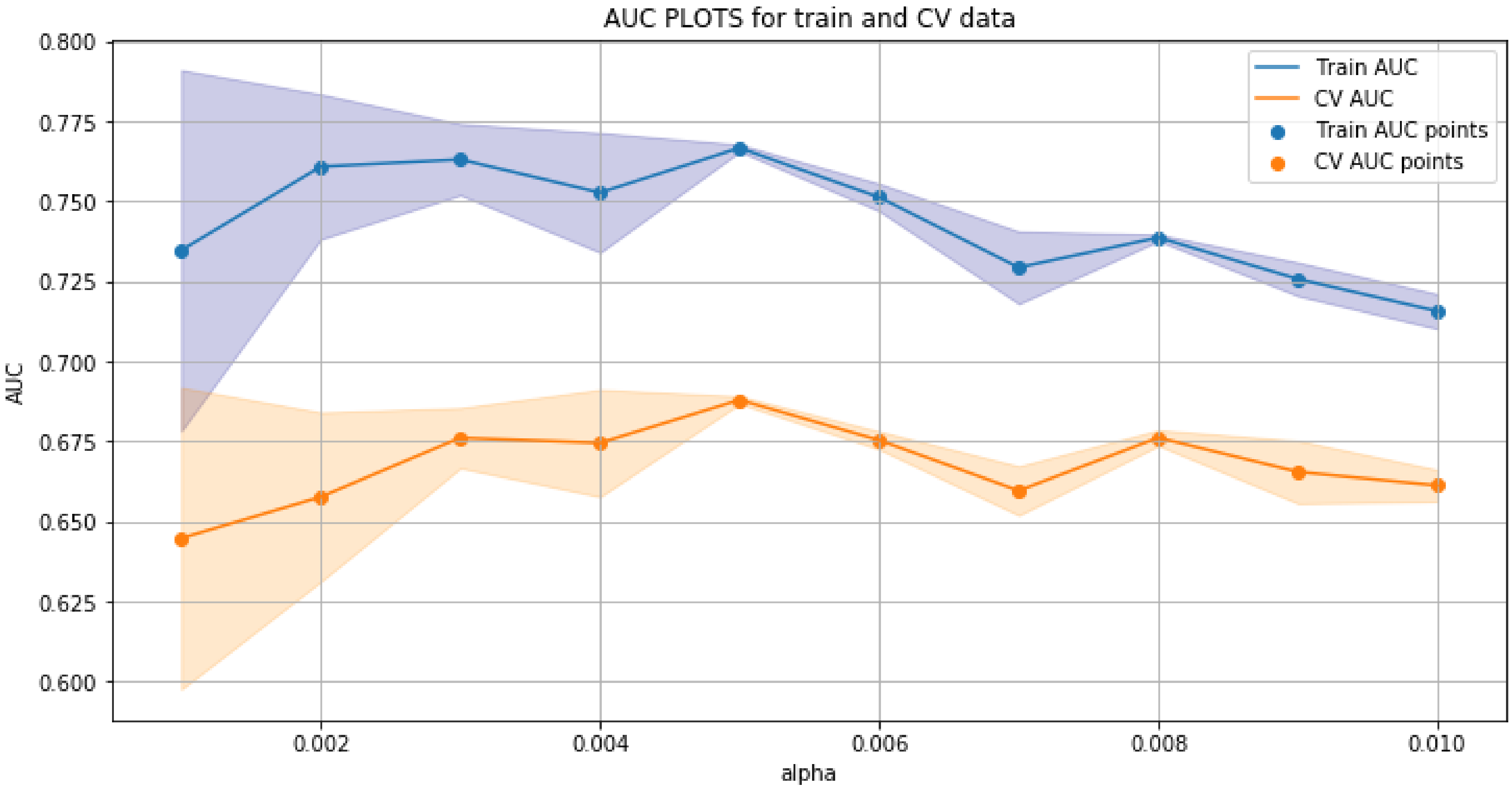
In [191]:
```
alphas = [10**i for i in range(-4, 5)]
auc_vs_K_plot(tfidf_train, y_train, alphas, penalty='l2', logplot=True)
```



Taking Range [0.001, 0.01]

In [193]:

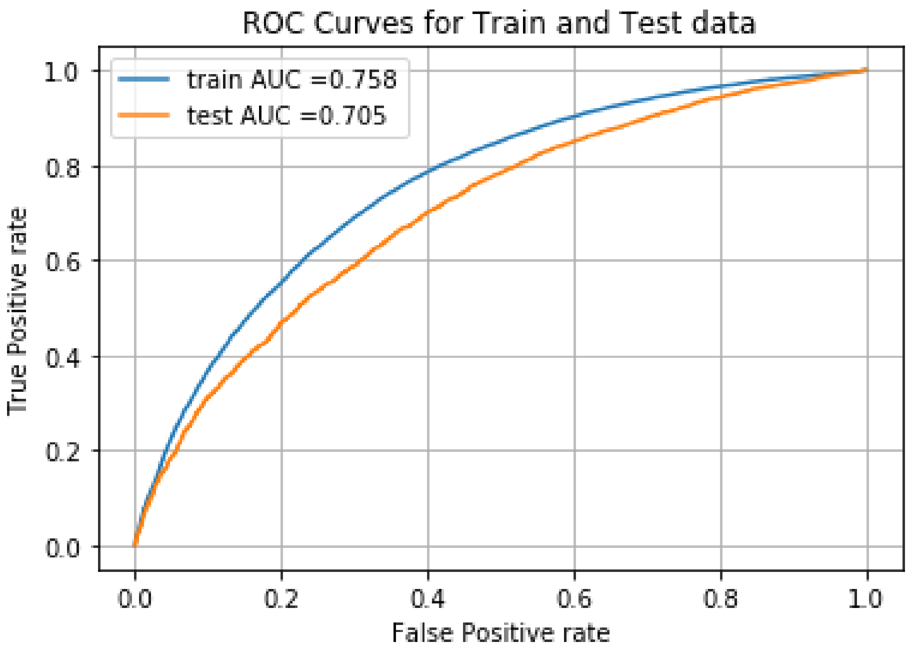
```
alphas = np.arange(0.001, 0.011, 0.001)
auc_vs_K_plot(tfidf_train, y_train, alphas, penalty='l2', logplot=False)
```



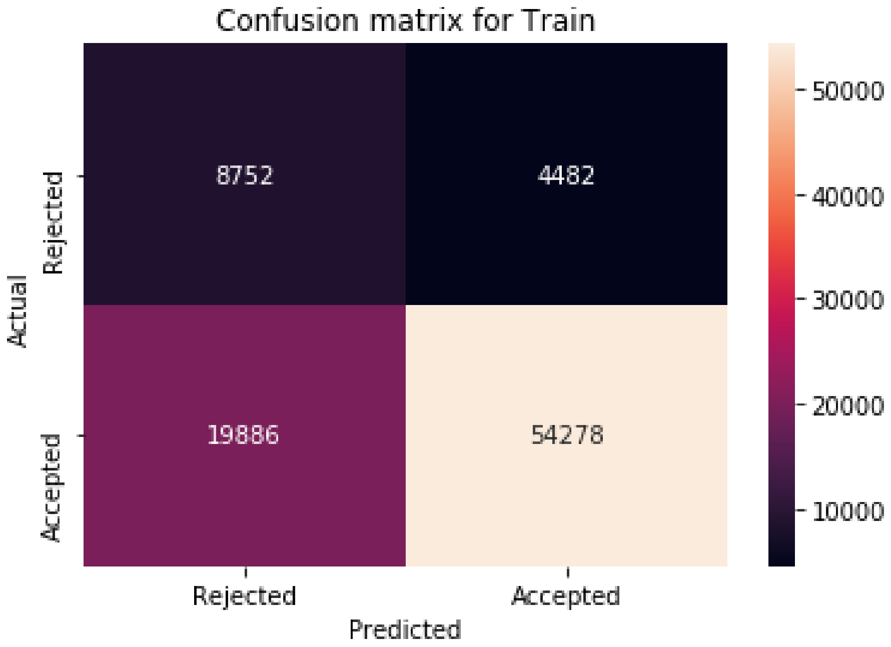
Taking best alpha = 0.005

```
In [194]: tfidf_l2_result = {}
          tfidf_l2_result[0.005] = ROC_conf_mat(tfidf_train, y_train, tfidf_test, y_test, 0.005, pena
```

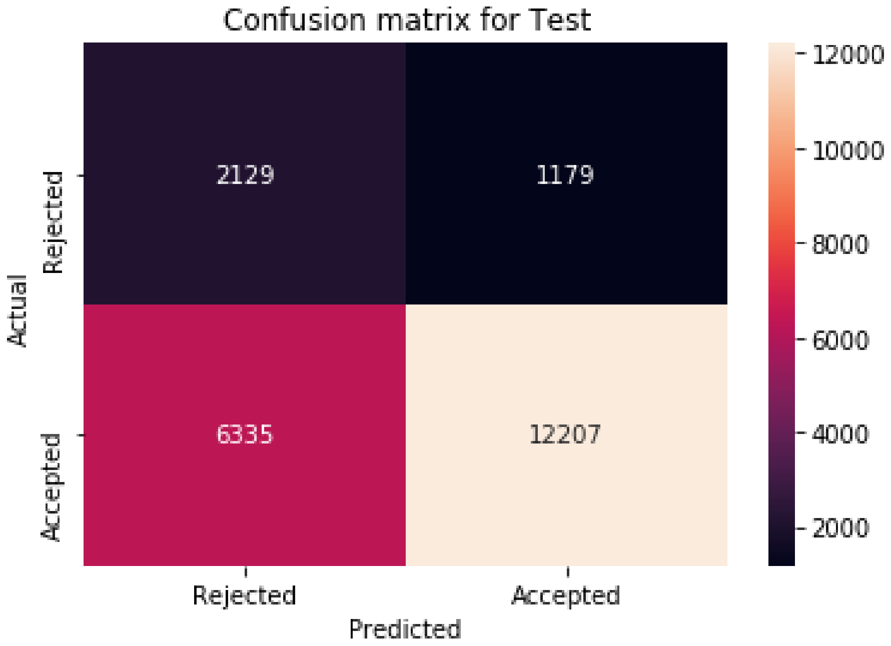
Analysis for alpha = 0.005



```
Confusion matrix for Train data with 0.8350296388340239 as threshold:
```

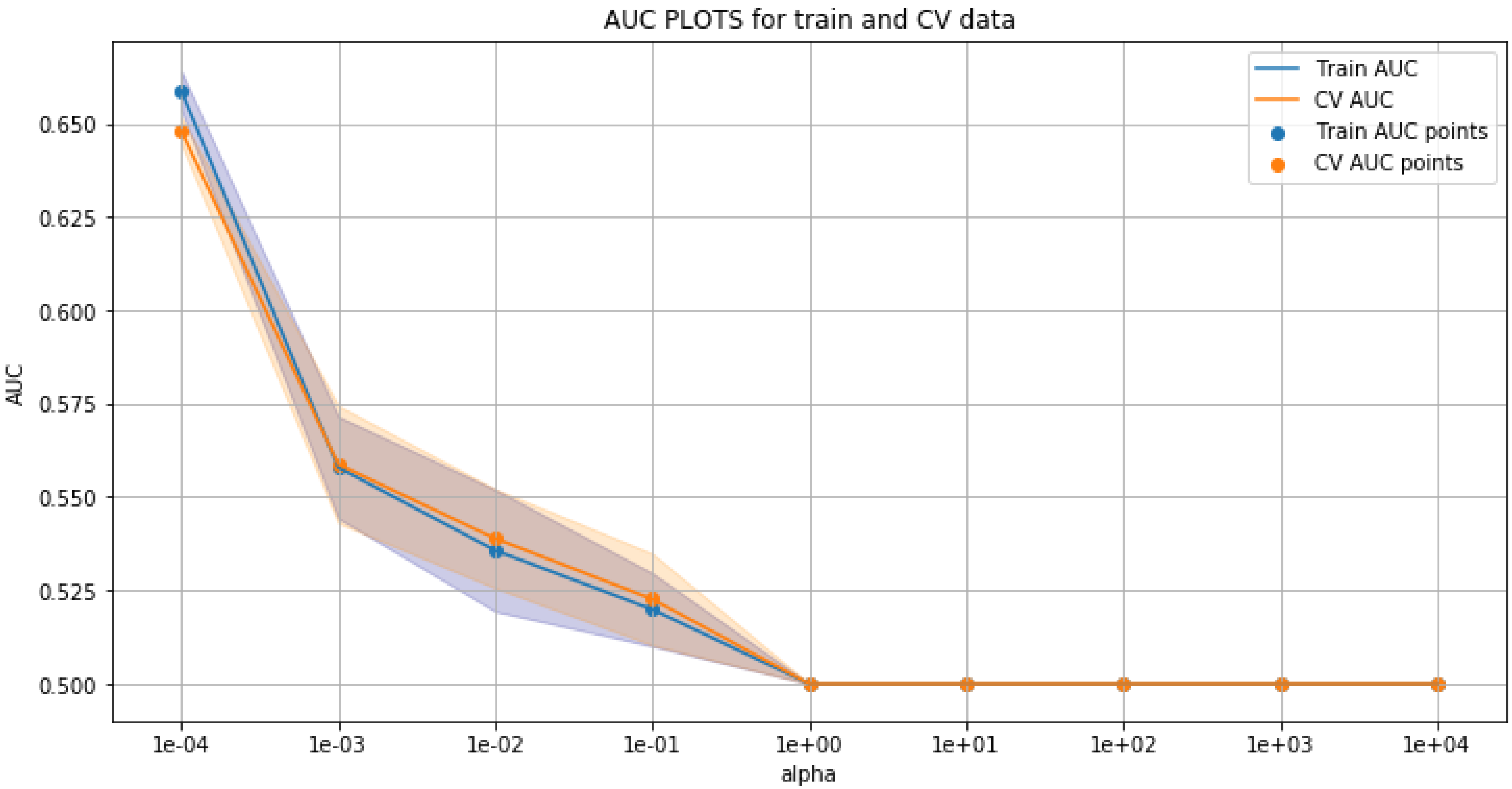


```
Confusion matrix for Test data with 0.8465387614454917 as threshold:
```



With L1 Penalty

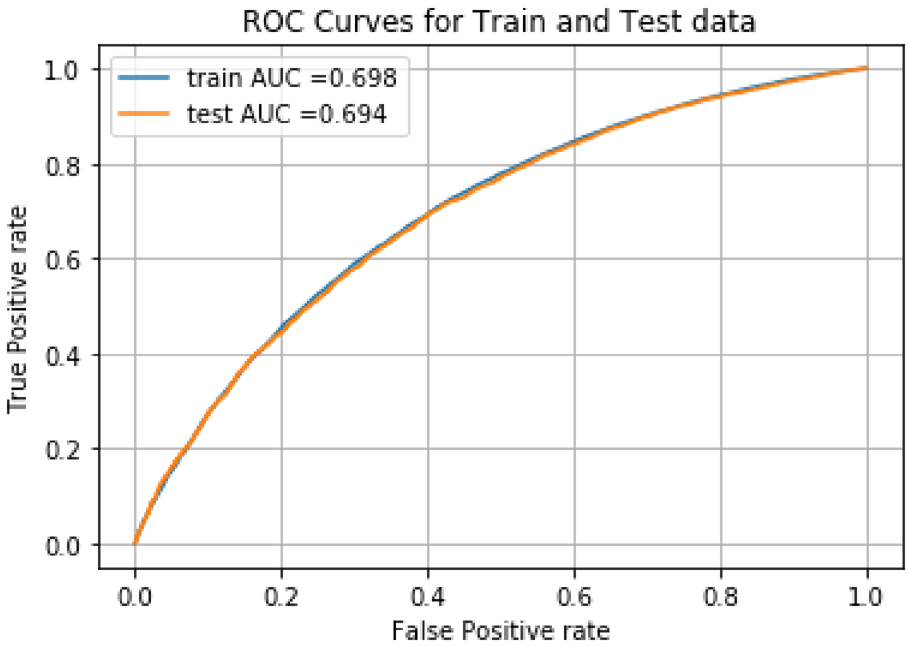
```
In [192]: alphas = [10**i for i in range(-4, 5)]
          auc_vs_K_plot(tfidf_train, y_train, alphas, penalty='l1', logplot=True)
```



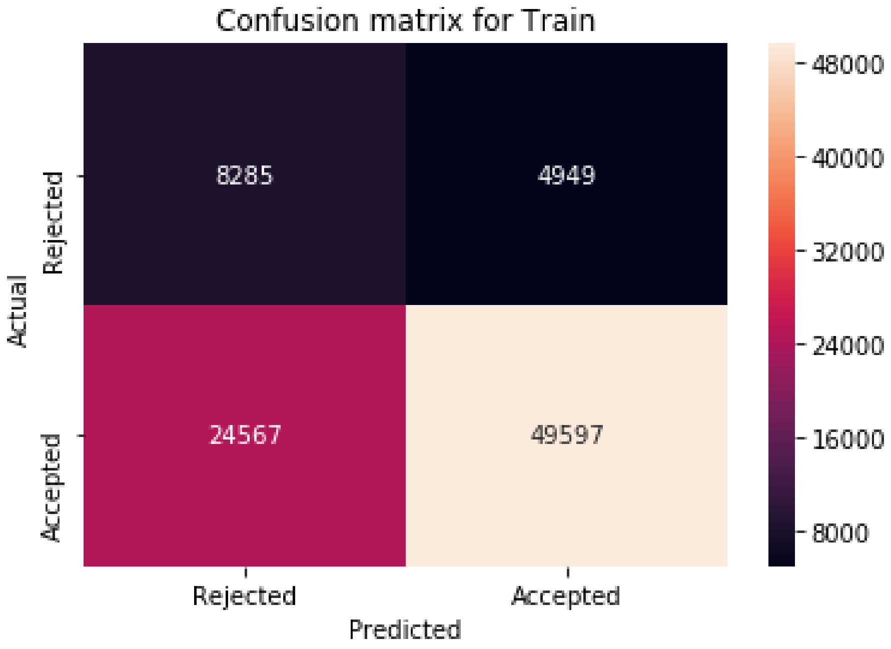
Taking 0.0001 as best alpha

In [195]:
```
tfidf_l1_result = {}
tfidf_l1_result[0.0001] = ROC_conf_mat(tfidf_train, y_train, tfidf_test, y_test, 0.0001, pe
```

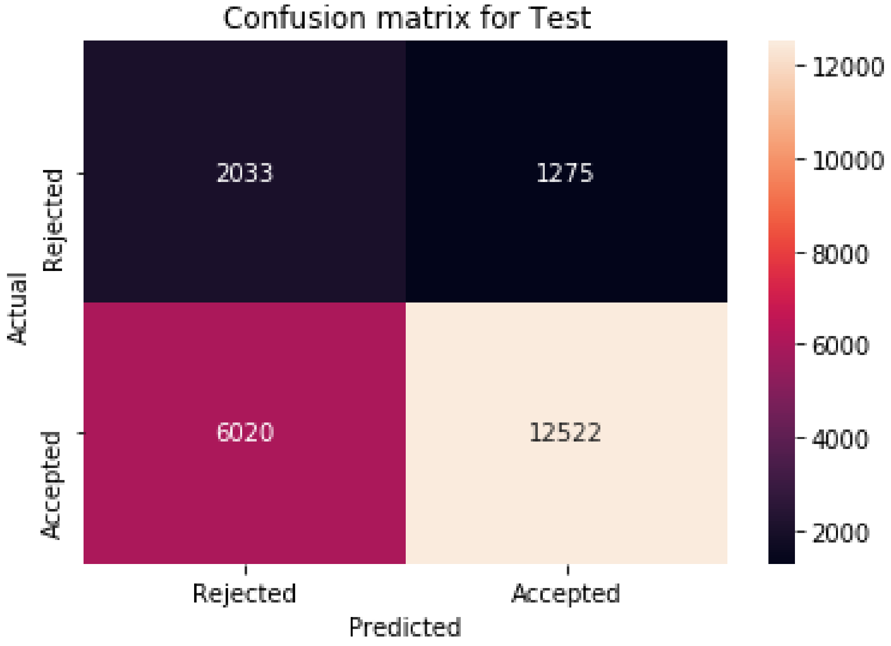
Analysis for alpha = 0.0001



```
Confusion matrix for Train data with 0.8390221845829777 as threshold:
```



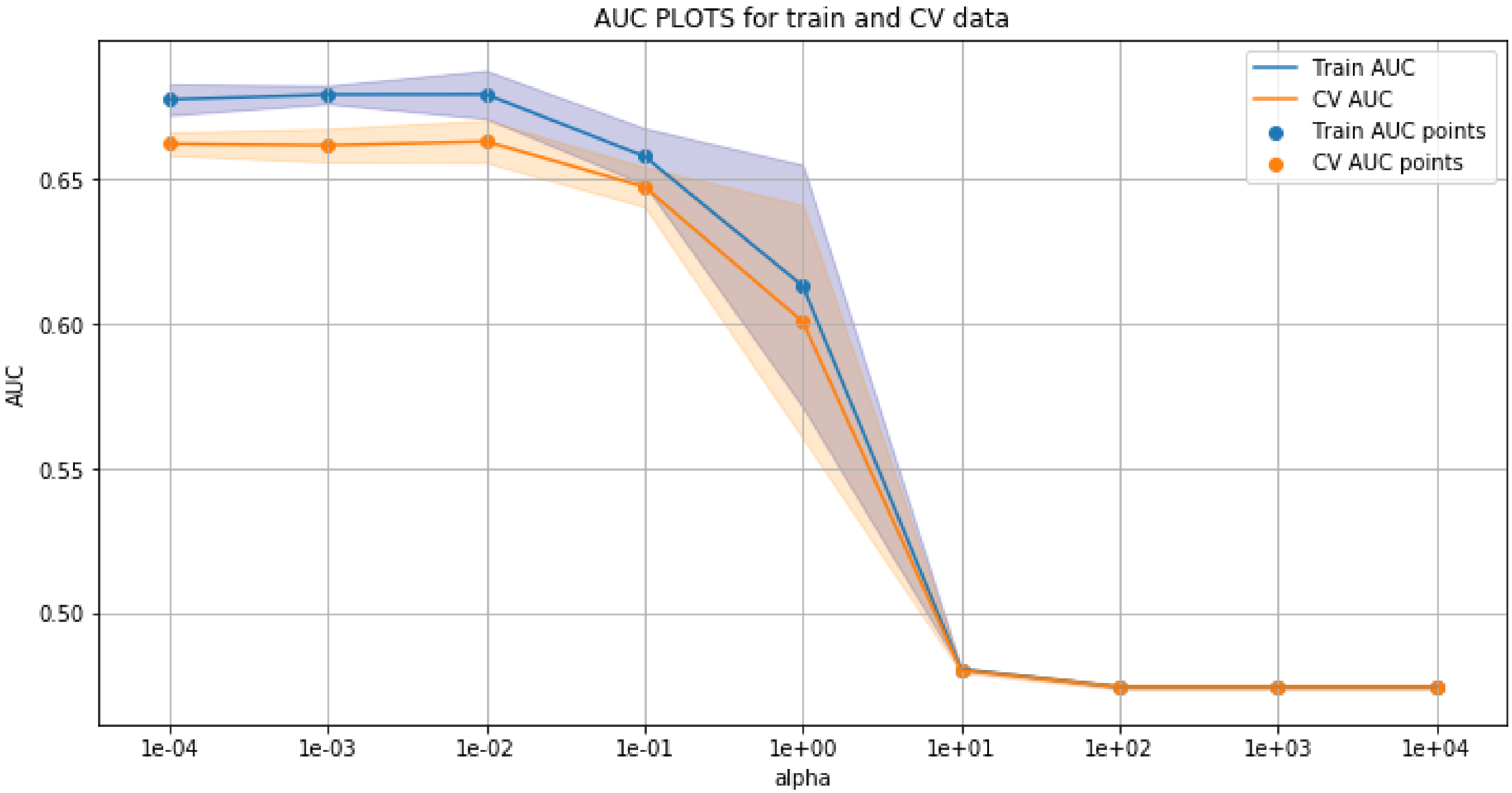
```
Confusion matrix for Test data with 0.8386831964778646 as threshold:
```



2.4.1 Applying SVM on Average Word2Vec, SET 3

With L2 Penalty

```
In [196]: alphas = [10**i for i in range(-4, 5)]
          auc_vs_K_plot(avgw2v_train, y_train, alphas, penalty='l2', logplot=True)
```



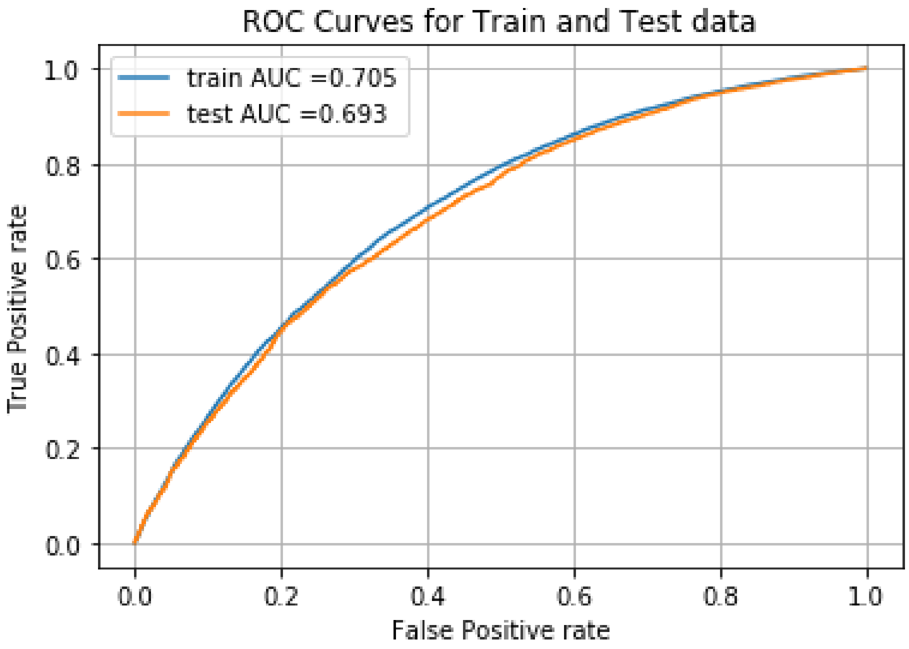
Taking best alpha = 0.0001

In [198]:

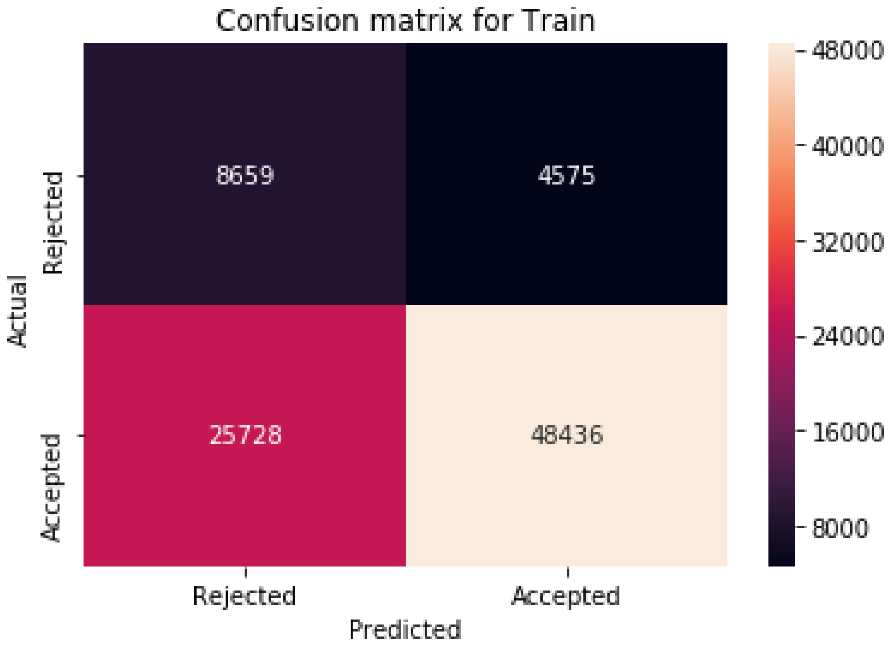
```
avgw2v_l2_result = {}
```

```
In [199]: avgw2v_l2_result[0.0001] = ROC_conf_mat(avgw2v_train, y_train, avgw2v_test, y_test, 0.0001,
```

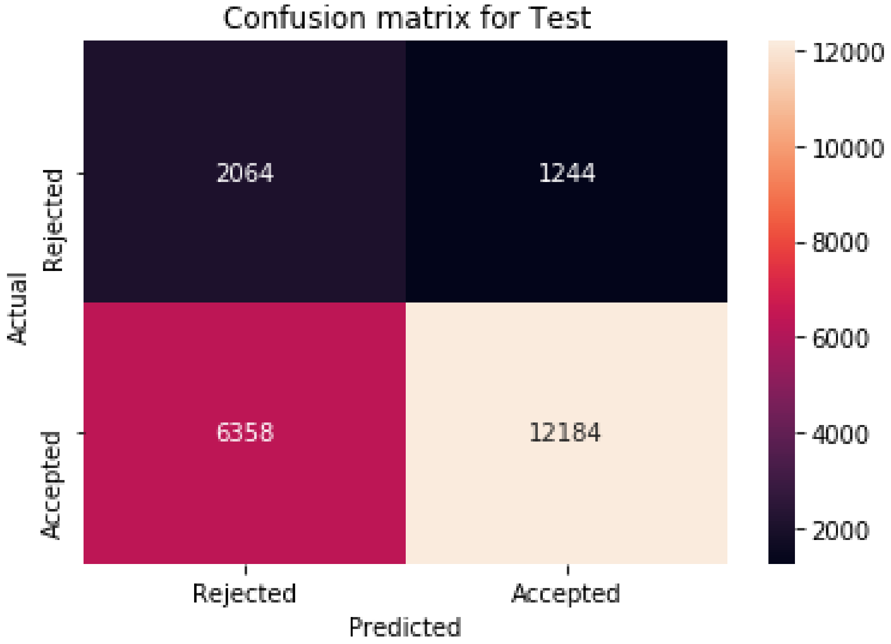
Analysis for alpha = 0.0001



```
Confusion matrix for Train data with 0.8414545542412878 as threshold:
```

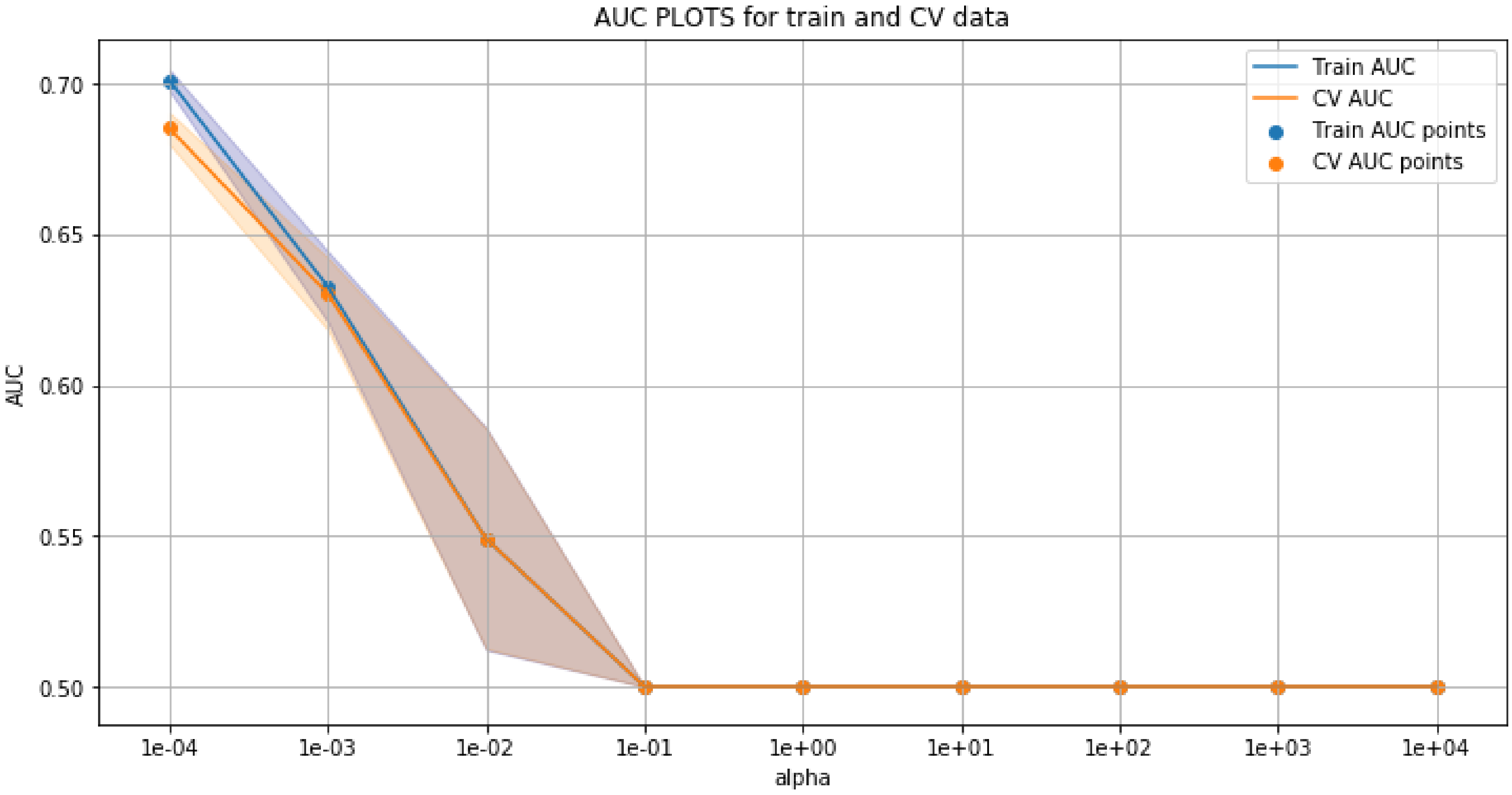


```
Confusion matrix for Test data with 0.8401334012873556 as threshold:
```



With L1 Penalty

In [197]:
```
alphas = [10**i for i in range(-4, 5)]
auc_vs_K_plot(avgw2v_train, y_train, alphas, penalty='l1', logplot=True)
```

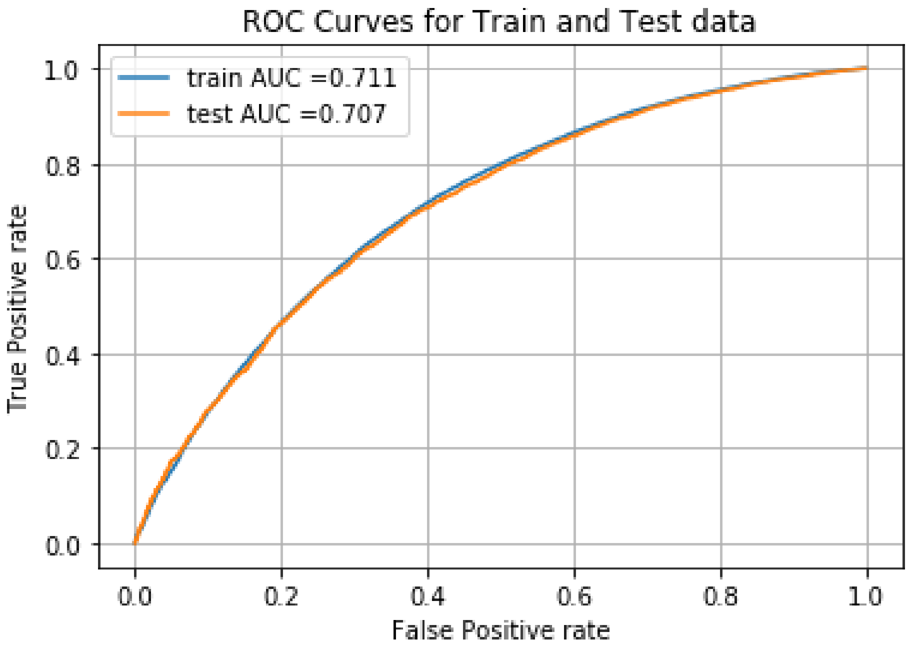


Taking best alpha = 0.0001

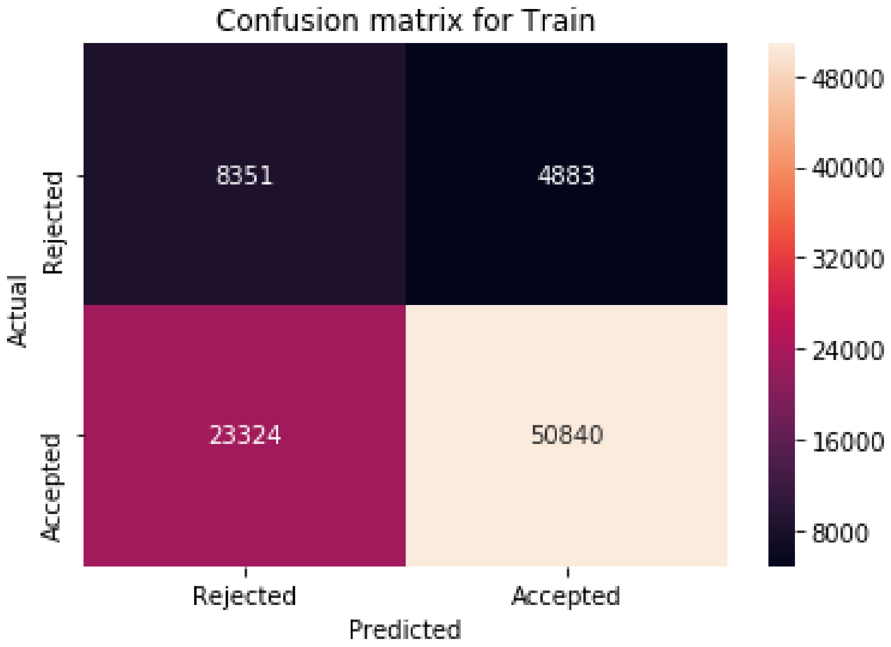
In [200]:
```
avgw2v_l1_result = {}
```

```
In [201]: avgw2v_l1_result[0.0001] = ROC_conf_mat(avgw2v_train, y_train, avgw2v_test, y_test, 0.0001,
```

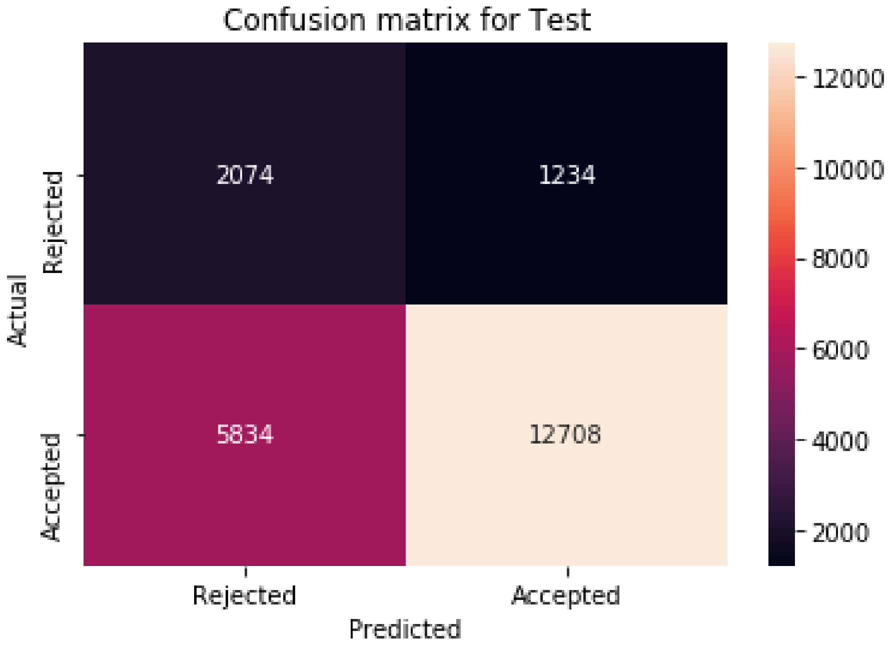
Analysis for alpha = 0.0001



```
Confusion matrix for Train data with 0.8339227066186595 as threshold:
```



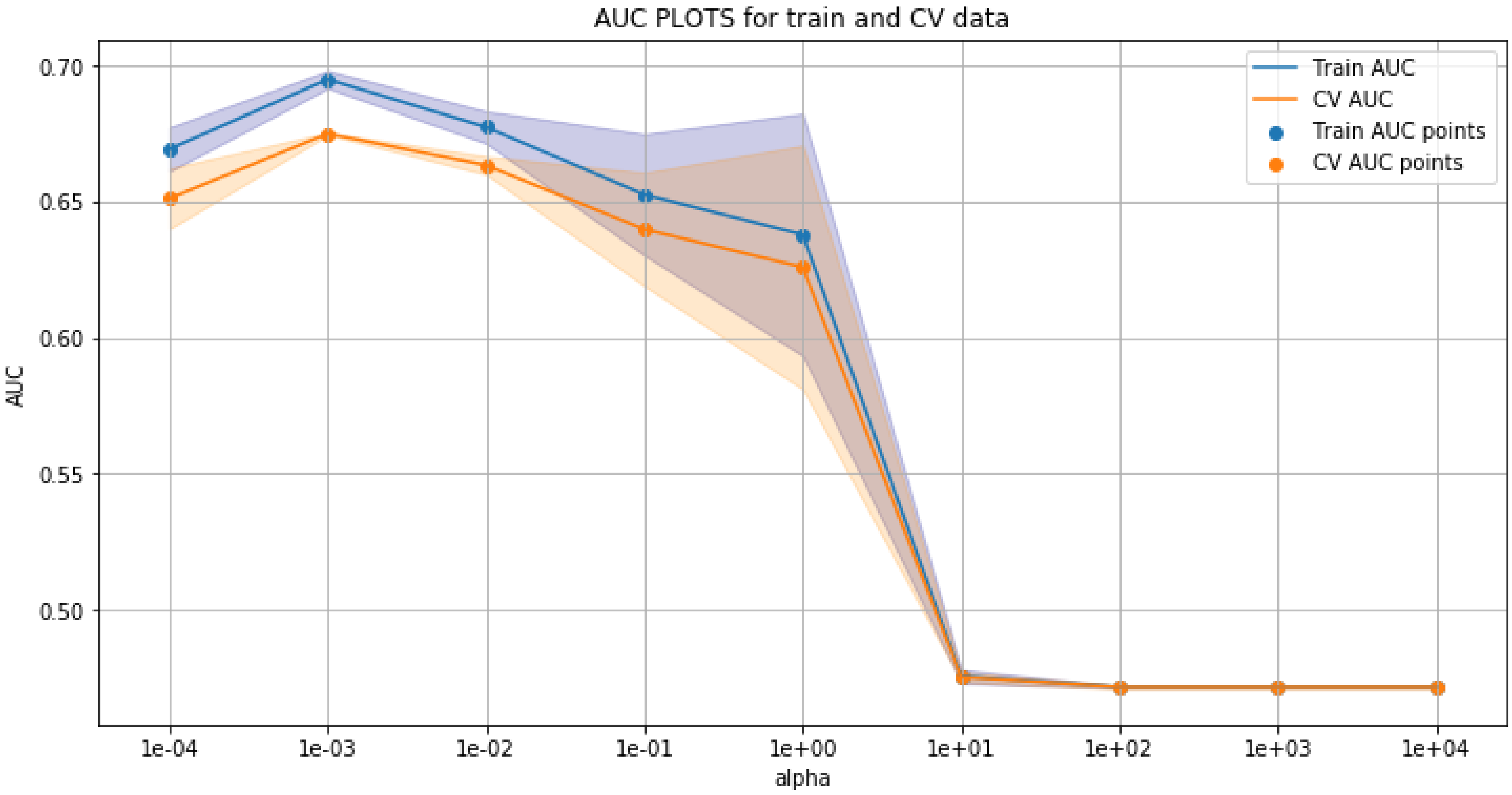
```
Confusion matrix for Test data with 0.8332486507643176 as threshold:
```



2.4.1 Applying SVM on TFIDF Weighted W2V, SET 4

With L2 Penalty

In [205]:
```
alphas = [10**i for i in range(-4, 5)]
auc_vs_K_plot(tfidfw2v_train, y_train, alphas, penalty='l2', logplot=True)
```



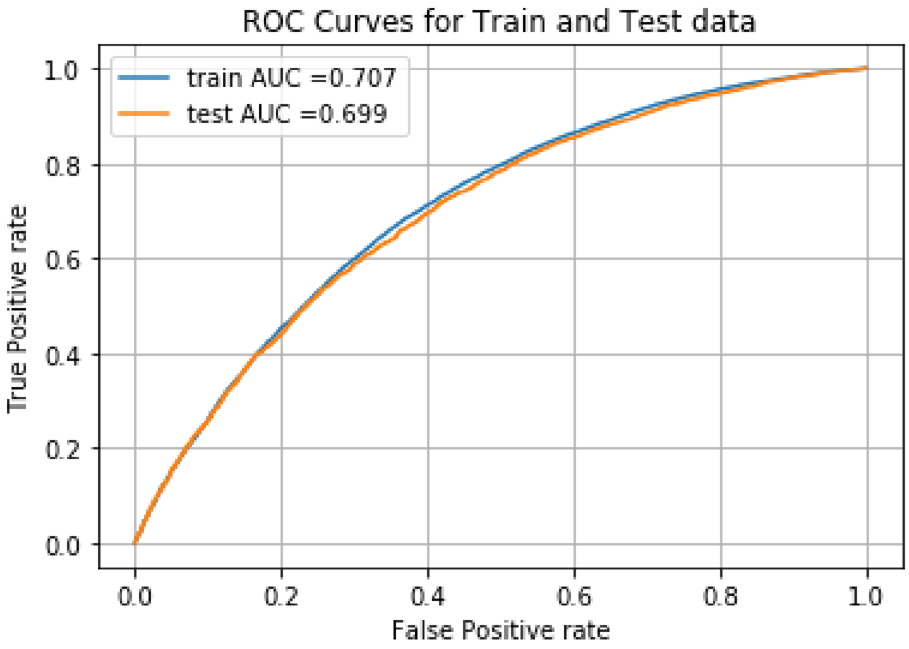
Taking best alpha = 0.001

In [207]:

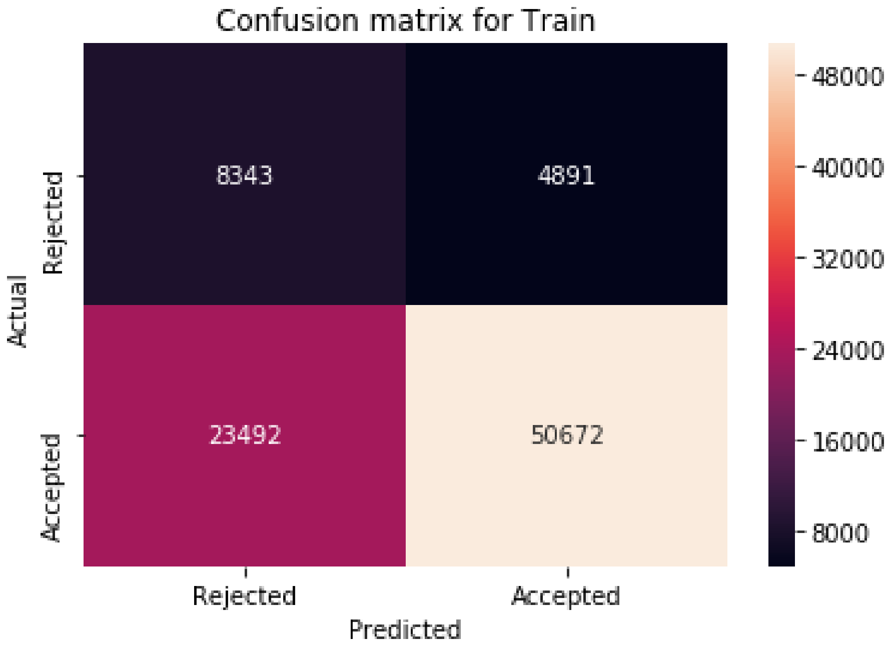
```
tfidfw2v_l2_result = {}
```

```
In [208]: tfidfw2v_l2_result[0.001] = ROC_conf_mat(tfidfw2v_train, y_train, tfidfw2v_test, y_test, 0.
```

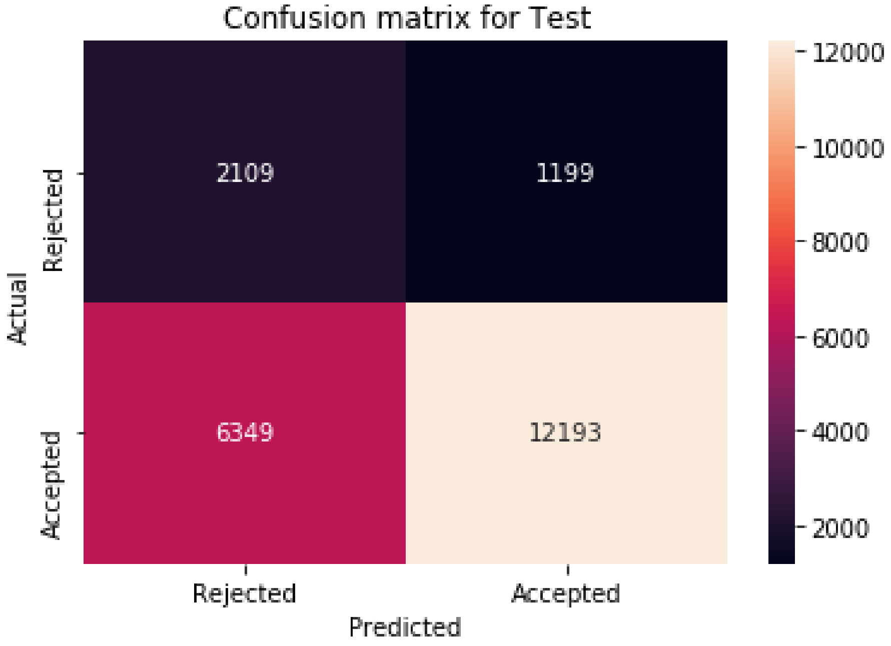
Analysis for alpha = 0.001



```
Confusion matrix for Train data with 0.8368130289044095 as threshold:
```

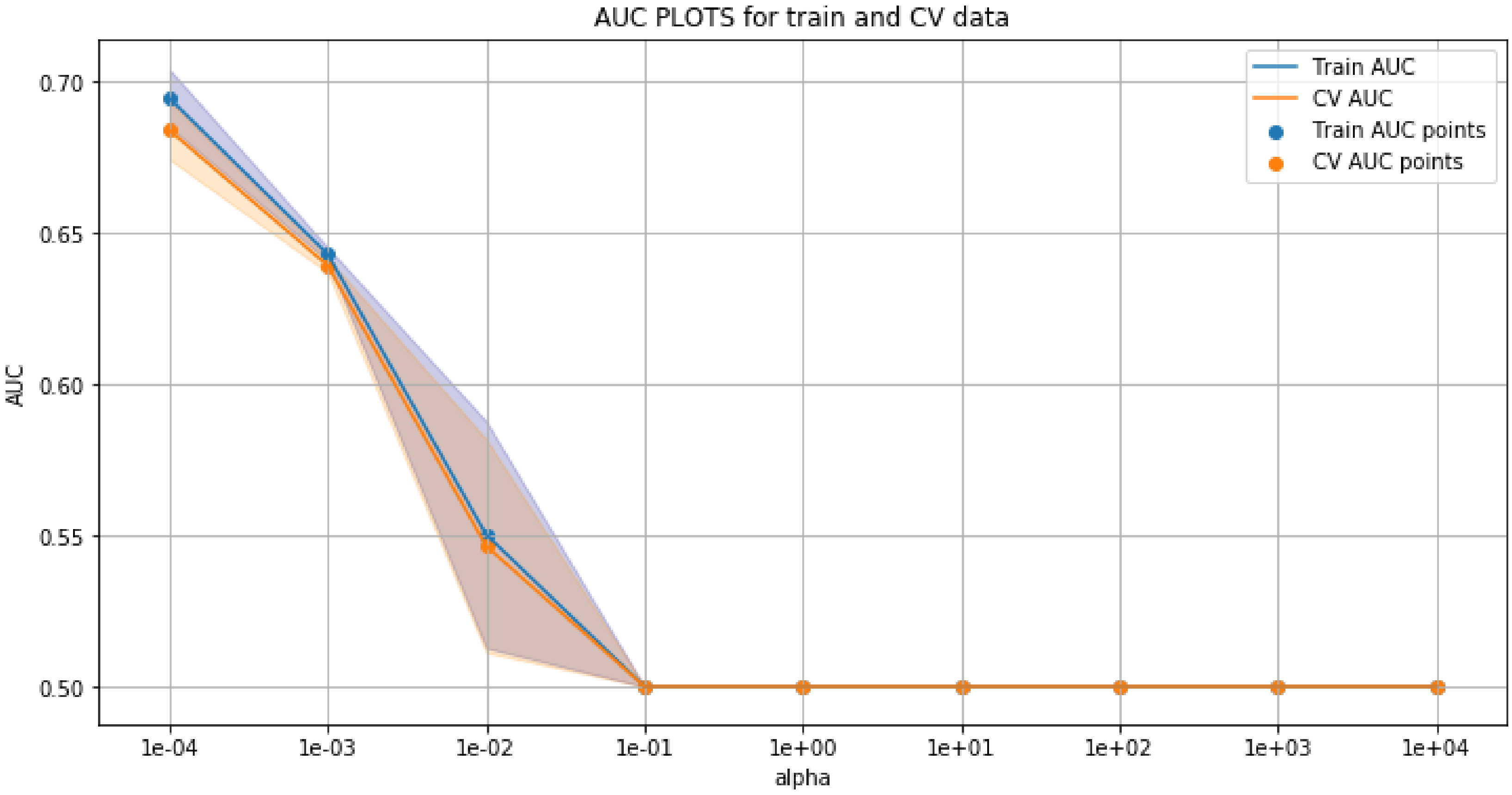


```
Confusion matrix for Test data with 0.8391071342251072 as threshold:
```



With L1 Penalty

```
In [206]: alphas = [10**i for i in range(-4, 5)]
          auc_vs_K_plot(tfidfw2v_train, y_train, alphas, penalty='l1', logplot=True)
```

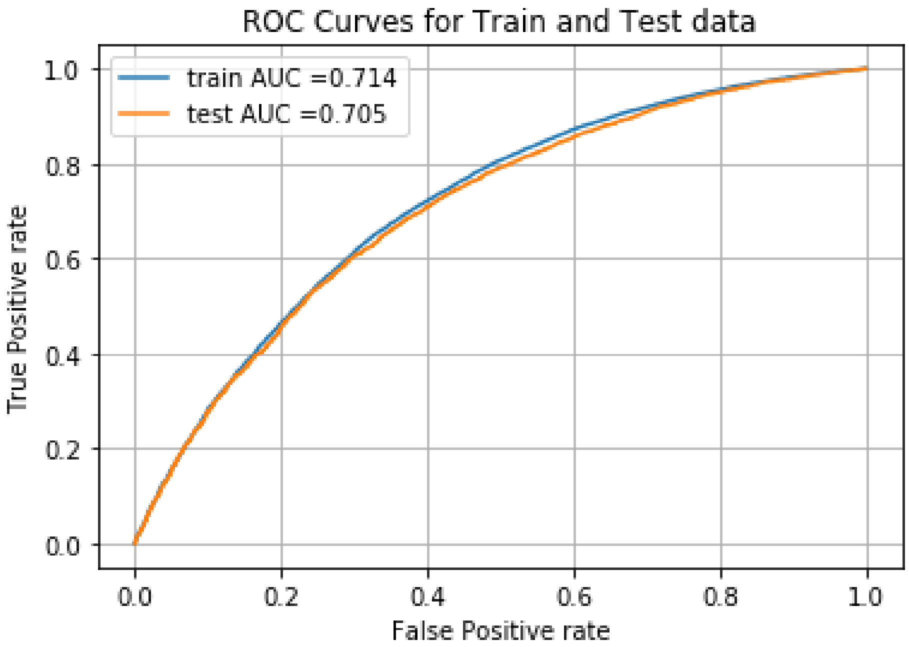


best alpha = 0.0001

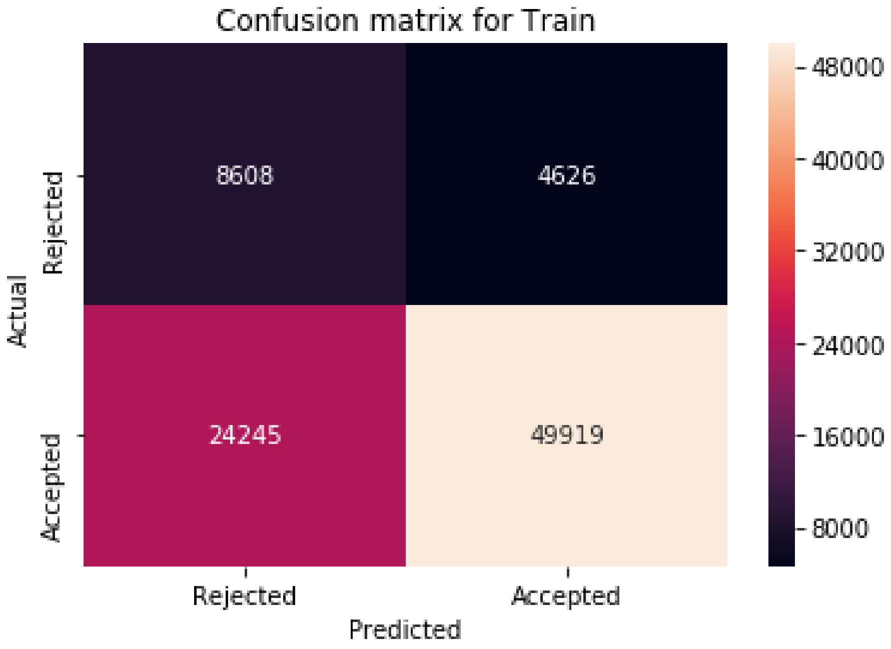
In [209]:

```
tfidfw2v_l1_result = {}
tfidfw2v_l1_result[0.0001] = ROC_conf_mat(tfidfw2v_train, y_train, tfidfw2v_test, y_test, 0
```

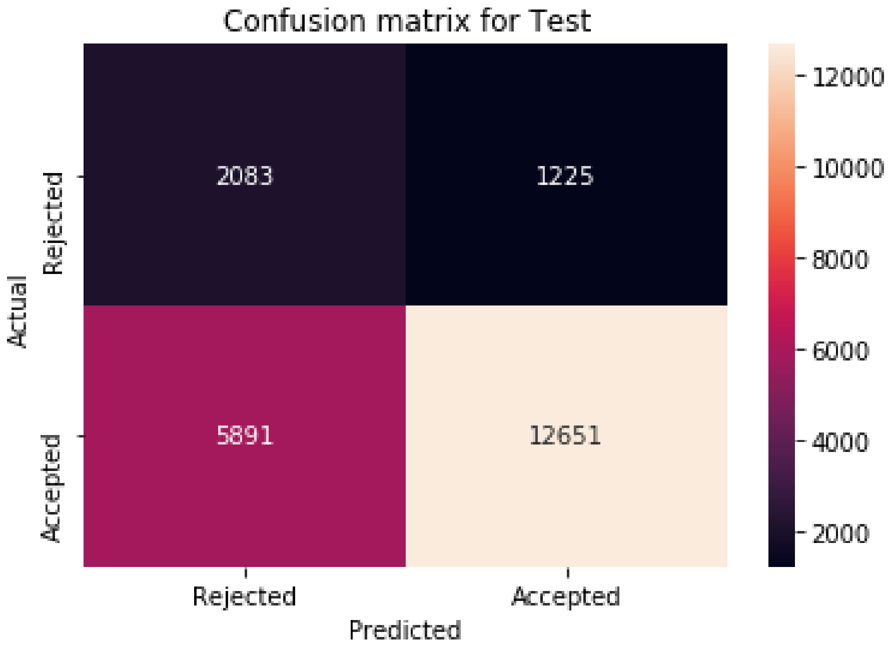
Analysis for alpha = 0.0001



```
Confusion matrix for Train data with 0.8387410813281586 as threshold:
```



```
Confusion matrix for Test data with 0.8345540099545289 as threshold:
```



2.5 Support Vector Machines with added Features `Set 5`

In [177]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Applying TruncatedSVD on TFIDF Vectors of train data that are created previously

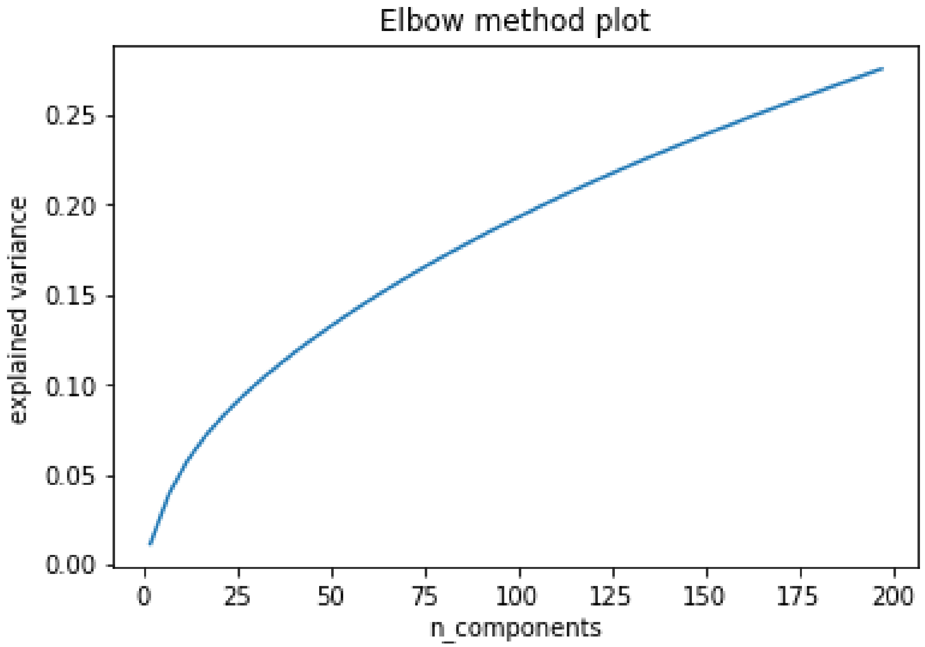
In [217]:

```
from sklearn.decomposition import TruncatedSVD
ncs = range(2, 201, 5)
variances = []
for nc in tqdm(ncs):
    svd = TruncatedSVD(n_components=nc)
    svd.fit(X_train_essay_tfidf)
    val = np.sum(svd.explained_variance_ratio_)
    variances.append(val)
```

...

In [218]:
```
plt.plot(ncs, variances)
plt.xlabel('n_components')
plt.ylabel('explained variance')
plt.title('Elbow method plot')
```

Out[218]:
```
Text(0.5,1,'Elbow method plot')
```



Appending some more values of n_components to see if explained variance will stop increasing at some point.

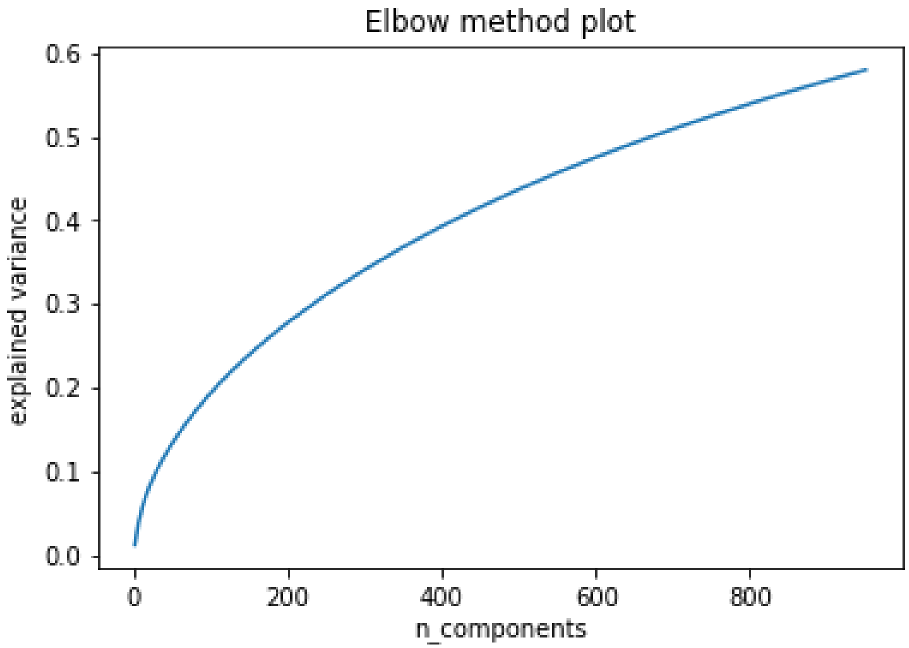
In [220]:
```
ncs = list(ncs)
```

```
In [221]: for nc in tqdm(range(250, 1000, 50)):
              svd = TruncatedSVD(n_components=nc)
              ncs.append(nc)
              svd.fit(X_train_essay_tfidf)
              val = np.sum(svd.explained_variance_ratio_)
              variances.append(val)
```

```
100%|████████████████████████████████████████████████████████████| 1
5/15 [25:58<00:00, 147.25s/it]
```

```
In [222]: plt.plot(ncs, variances)
          plt.xlabel('n_components')
          plt.ylabel('explained variance')
          plt.title('Elbow method plot')
```

```
Out[222]: Text(0.5,1,'Elbow method plot')
```



Elbow seems to be at n_components = 25 but taking a higher value to get good variance. So n_components = 200.

In [223]:

```
svd = TruncatedSVD(n_components=200)
svd.fit(X_train_essay_tfidf)
X_train_essay_svd = svd.transform(X_train_essay_tfidf)
X_test_essay_svd = svd.transform(X_test_essay_tfidf)
print(X_train_essay_svd.shape, X_test_essay_svd.shape)
```

```
(87398, 200) (21850, 200)
```

Getting sentiment scores for different essays and numerical and categorical features to produce our input matrix for set-5

In [224]:

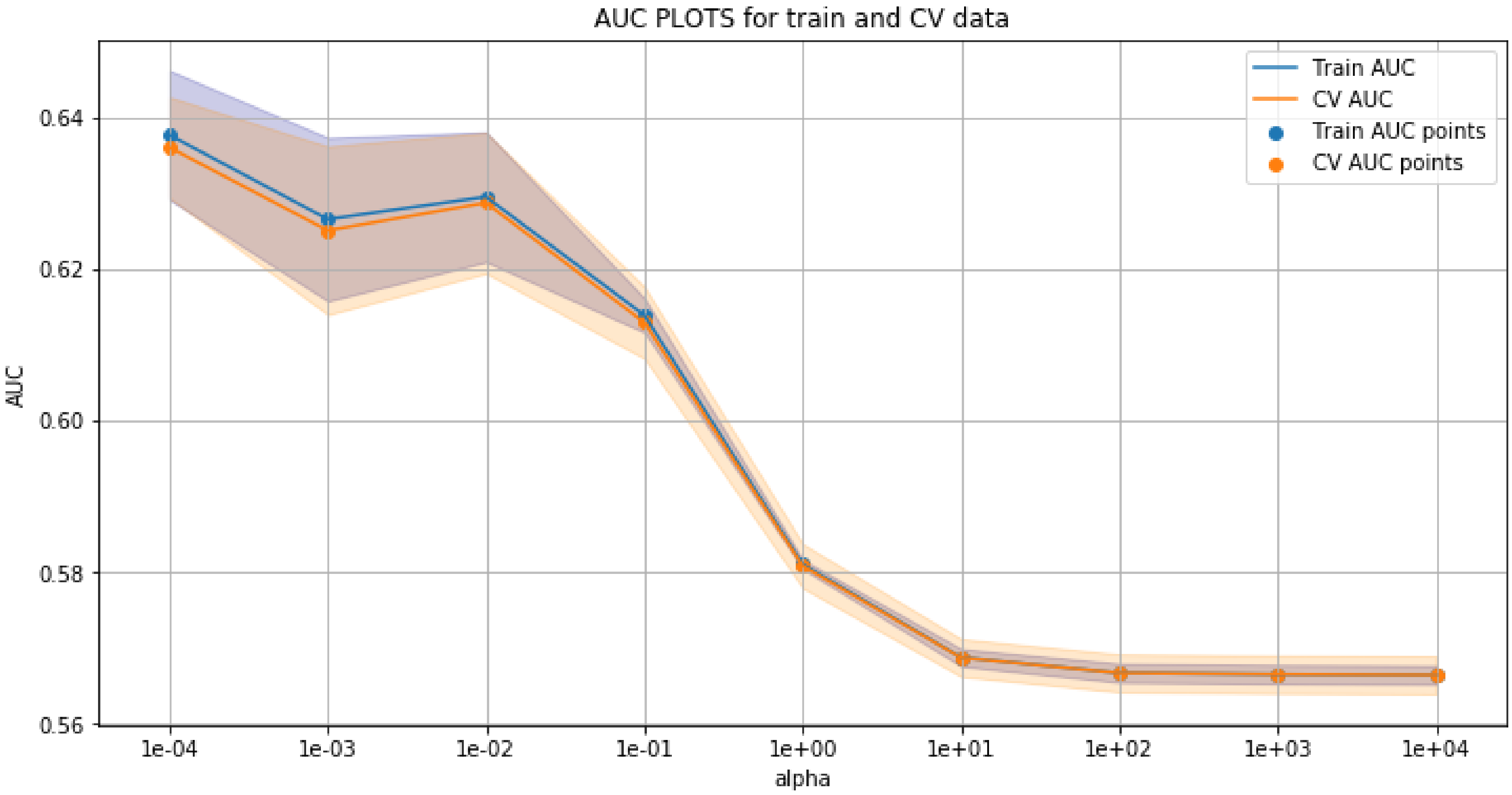
```
sentim_cols = ['essay1_neg', 'essay1_nue', 'essay1_pos', 'essay1_comp', 'essay2_neg',\
       'essay2_nue', 'essay2_pos', 'essay2_comp', 'essay1_neg', 'essay1_nue',\
       'essay1_pos', 'essay1_comp', 'essay2_neg', 'essay2_nue', 'essay2_pos',\
       'essay2_comp', 'essay3_neg', 'essay3_nue', 'essay3_pos', 'essay3_comp',\
       'essay4_neg', 'essay4_nue', 'essay4_pos', 'essay4_comp', 'essay_word_count', 'title_
Task2_train = hstack((cat_num_train, np.array(X_train[sentim_cols]), X_train_essay_svd))
Task2_test = hstack((cat_num_test, np.array(X_test[sentim_cols]), X_test_essay_svd))

print(Task2_train.shape, y_train.shape)
print(Task2_test.shape, y_test.shape)
```

```
(87398, 329) (87398,)
(21850, 329) (21850,)
```

With L2 Penalty

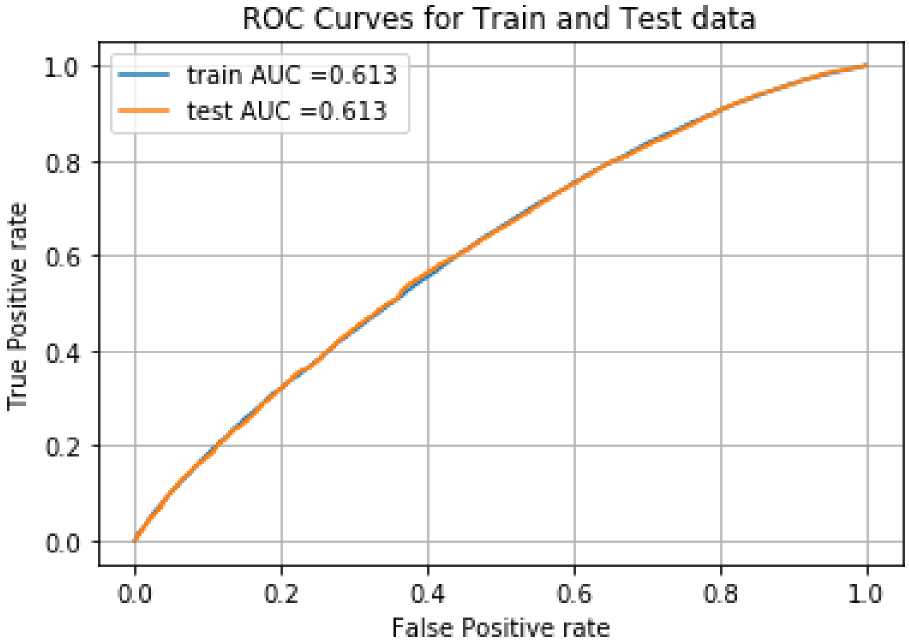
```
In [226]: alphas = [10**i for i in range(-4, 5)]
          auc_vs_K_plot(Task2_train, y_train, alphas, penalty='l2', logplot=True)
```



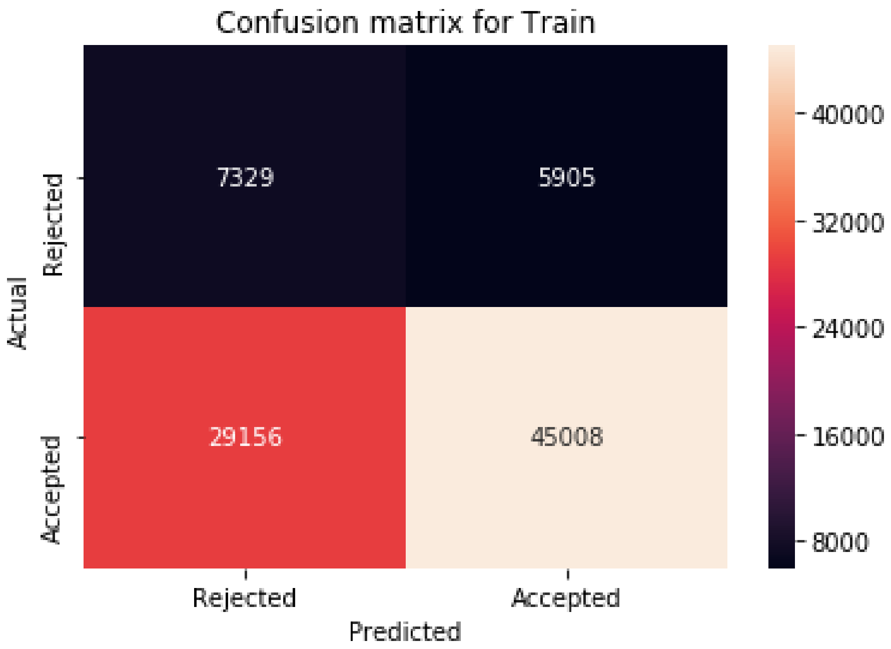
best alpha = 0.0001

In [228]:
```
task2_l2_result = {}
task2_l2_result[0.0001] = ROC_conf_mat(Task2_train, y_train, Task2_test, y_test, 0.0001, pe
```

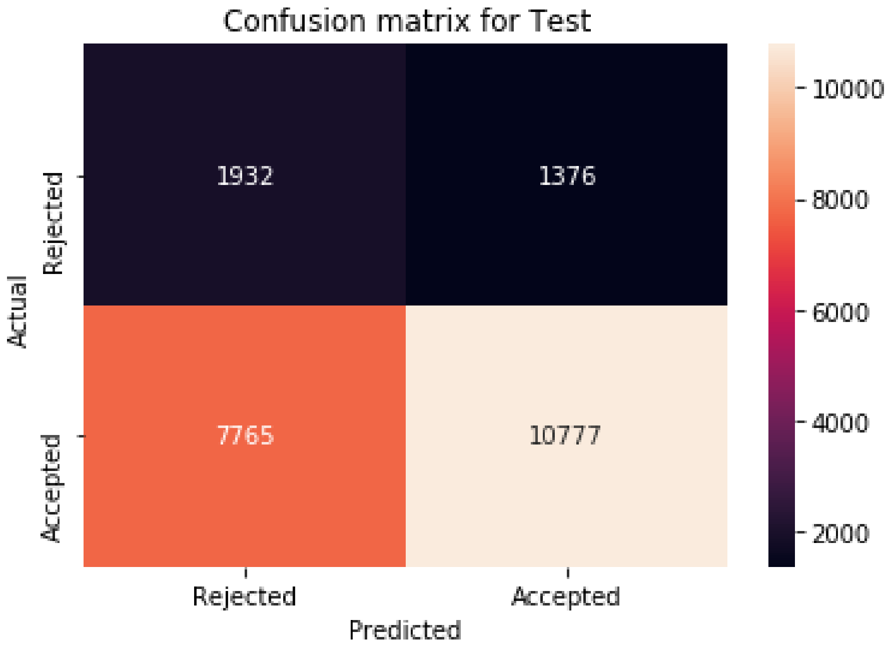
Analysis for alpha = 0.0001



```
Confusion matrix for Train data with 0.8387504732411609 as threshold:
```

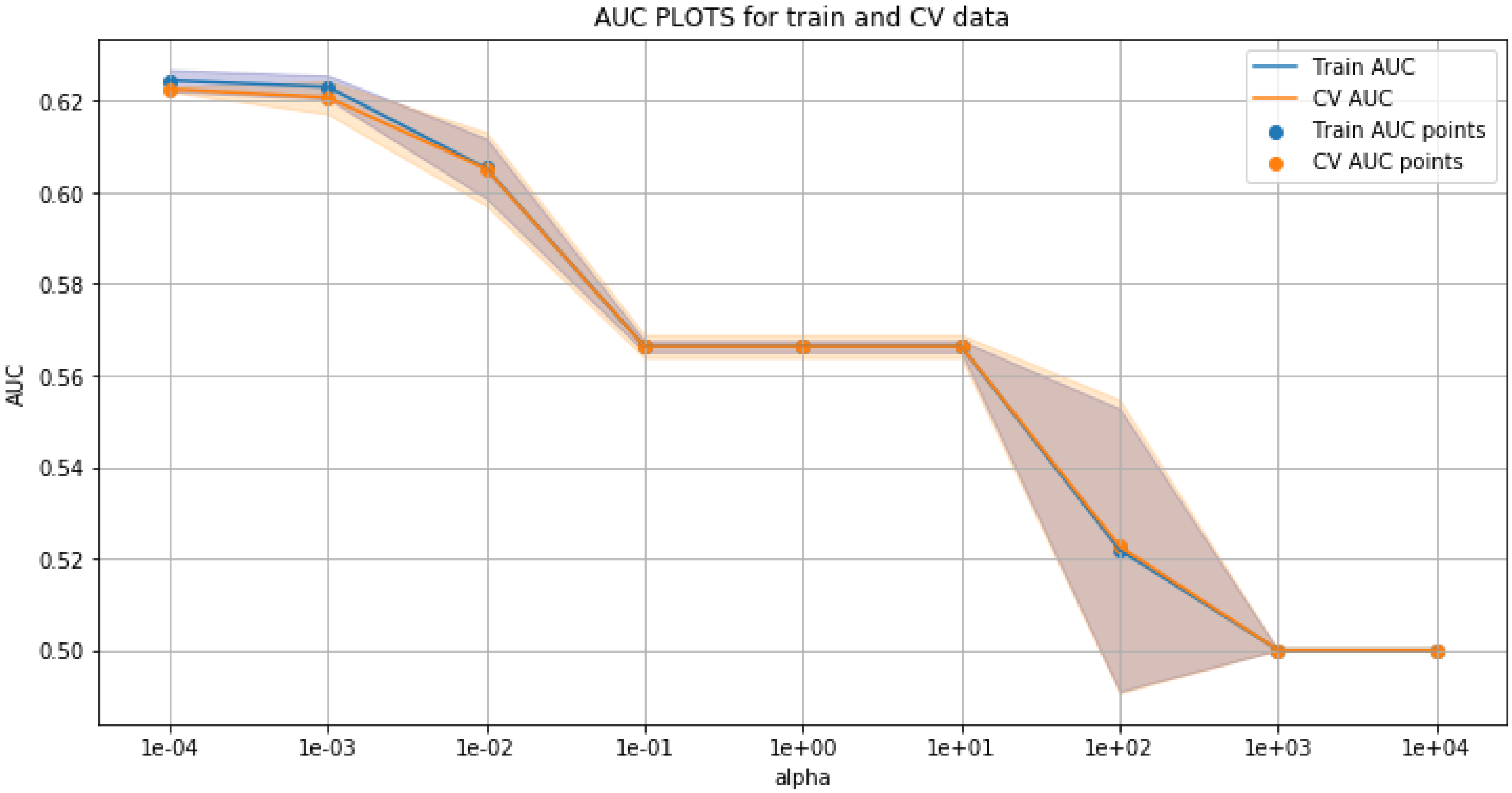


```
Confusion matrix for Test data with 0.8408675307760971 as threshold:
```



With L1 Penalty

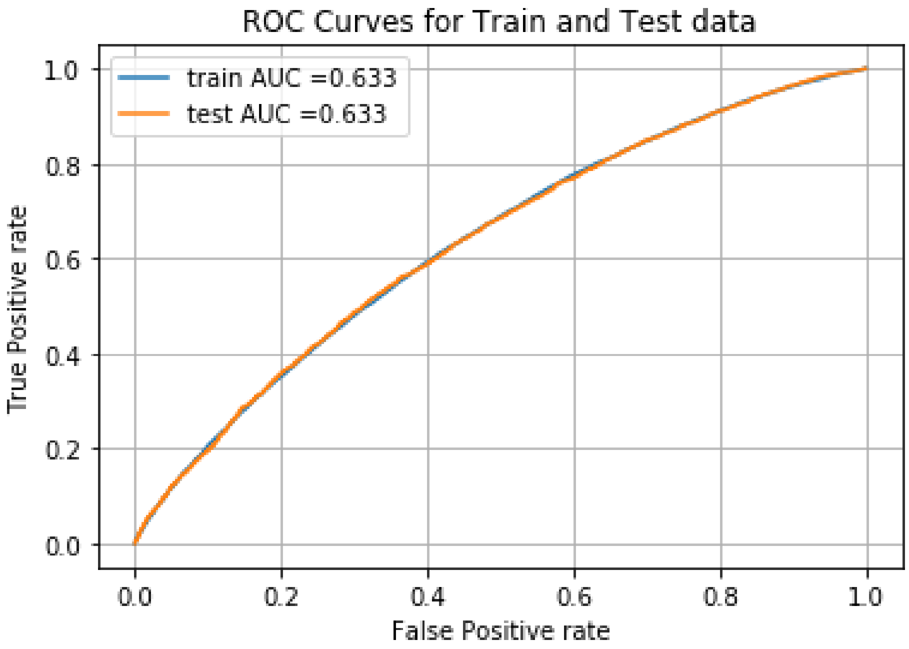
```
In [227]: alphas = [10**i for i in range(-4, 5)]
          auc_vs_K_plot(Task2_train, y_train, alphas, penalty='l1', logplot=True)
```



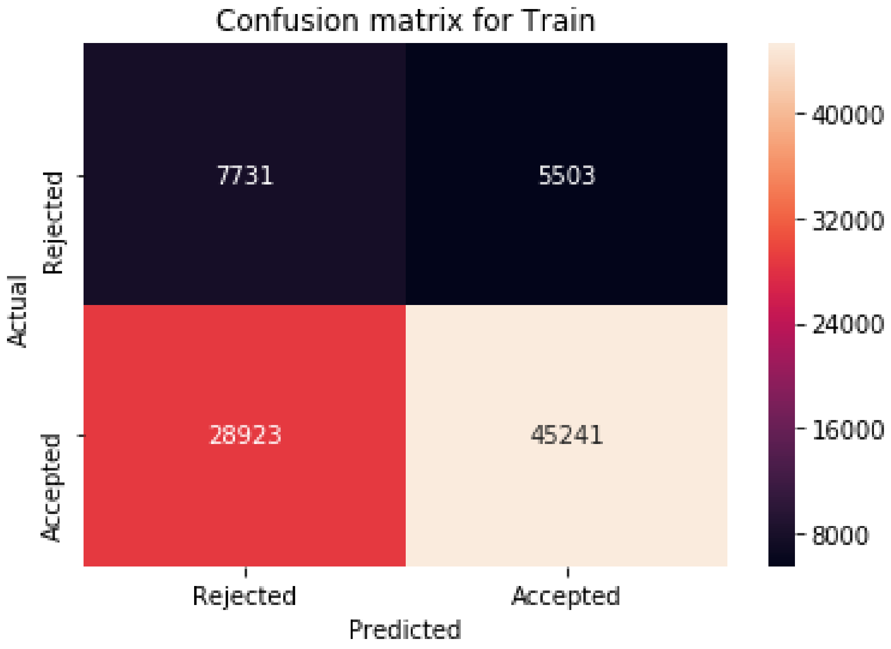
best alpha = 0.0001

In [229]:
```
task2_l1_result = {}
task2_l1_result[0.0001] = ROC_conf_mat(Task2_train, y_train, Task2_test, y_test, 0.0001, pe
```

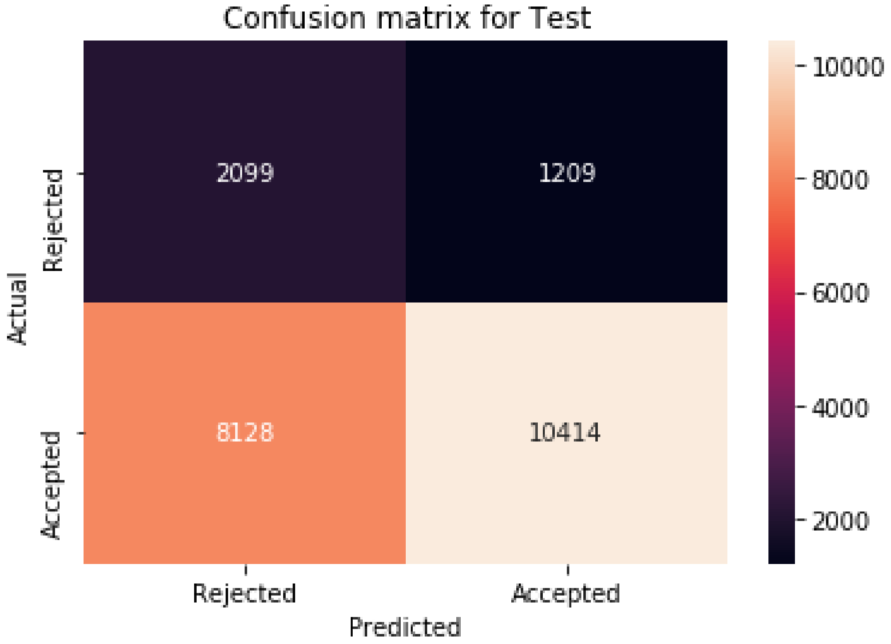
Analysis for alpha = 0.0001



```
Confusion matrix for Train data with 0.8364665143594445 as threshold:
```



```
Confusion matrix for Test data with 0.843031873284791 as threshold:
```



3. Conclusion

Summarising the results

In [241]:

```
task2_l1_result
```

Out[241]:

```
{0.0001: {'train_auc': 0.6332853977384196, 'test_auc': 0.633253151086647}}
```

In [244]:

```
from prettytable import PrettyTable
table = PrettyTable()
table.field_names = ['SET', 'alpha','Penalty', 'Train AUC', 'Test AUC']
table.add_row(['Bag of Words', 0.02, 'l2', np.round(bow_l2_result[0.02]['train_auc'], 3),\
               np.round(bow_l2_result[0.02]['test_auc'], 3)])
table.add_row(['Bag of Words', 0.0008, 'l1', np.round(bow_l1_result[0.0008]['train_auc'], 3
               np.round(bow_l1_result[0.0008]['test_auc'], 3)])
table.add_row(['TfIdf', 0.005, 'l2', np.round(tfidf_l2_result[0.005]['train_auc'], 3),\
               np.round(tfidf_l2_result[0.005]['test_auc'], 3)])
table.add_row(['TfIdf', 0.0001, 'l1', np.round(tfidf_l1_result[0.0001]['train_auc'], 3),\
               np.round(tfidf_l1_result[0.0001]['test_auc'], 3)])
table.add_row(['Average Word2Vec', 0.0001, 'l2', np.round(avgw2v_l2_result[0.0001]['train_a
               np.round(avgw2v_l2_result[0.0001]['test_auc'], 3)])
table.add_row(['Average Word2Vec', 0.0001, 'l1', np.round(avgw2v_l1_result[0.0001]['train_a
               np.round(avgw2v_l1_result[0.0001]['test_auc'], 3)])
table.add_row(['TfIdf Word2Vec', 0.001, 'l2', np.round(tfidfw2v_l2_result[0.001]['train_auc
               np.round(tfidfw2v_l2_result[0.001]['test_auc'], 3)])
table.add_row(['TfIdf Word2Vec', 0.0001, 'l1', np.round(tfidfw2v_l1_result[0.0001]['train_a
               np.round(tfidfw2v_l1_result[0.0001]['test_auc'], 3)])
table.add_row(['Task 2 data', 0.0001, 'l2', np.round(task2_l2_result[0.0001]['train_auc'],
               np.round(task2_l2_result[0.0001]['test_auc'], 3)])
table.add_row(['Task 2 data', 0.0001, 'l1', np.round(task2_l1_result[0.0001]['train_auc'],
               np.round(task2_l1_result[0.0001]['test_auc'], 3)])
print(table)
```

```
+------------------+--------+---------+-----------+----------+
|       SET        | alpha  | Penalty | Train AUC | Test AUC |
+------------------+--------+---------+-----------+----------+
|   Bag of Words   |  0.02  |    l2   |   0.744   |  0.701   |
|   Bag of Words   | 0.0008 |    l1   |   0.671   |  0.663   |
|      TfIdf       | 0.005  |    l2   |   0.758   |  0.705   |
|      TfIdf       | 0.0001 |    l1   |   0.698   |  0.694   |
| Average Word2Vec | 0.0001 |    l2   |   0.705   |  0.693   |
| Average Word2Vec | 0.0001 |    l1   |   0.711   |  0.707   |
|  TfIdf Word2Vec  | 0.001  |    l2   |   0.707   |  0.699   |
|  TfIdf Word2Vec  | 0.0001 |    l1    |   0.714    |  0.705    |
|   Task 2 data    | 0.0001 |    l2    |   0.613    |  0.613    |
|   Task 2 data    | 0.0001 |    l1    |   0.633    |  0.633    |
+------------------+--------+---------+-----------+----------+
```

Conclusion:

- **We can see Both Word2Vec models did good with Both L2 and L1 Regularization. And The performance is also high with Bow and Tfidf with L2 Regularization.**
- **Using only numerical, categorical and Truncated SVD data didnt do much good. The performance is lower than all other models. If time performance is very important Word2Vec can be taken as it has much less columns than BOW and TFIDF models.**
- **Compared to previous models i.e. KNN, Logistic Regerssion etc.. Linear SVM didnt do better than Logistic regression. But did better than other Models. And the gap between Train and Test performance reduced when compared with logistic regression. So we can say Linear SVM is less over-fitting than Logistic Regression.**

```
In [ ]:
```