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	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`

`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* 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 [1]:

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

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

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

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
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
C:\Users\Public\Anaconda3\lib\site-packages\ge
nsim\utils.py:1197: UserWarning: detected Wind
ows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing ch
unkize to chunkize_serial")
```

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('C:/Users/pramod reddy chandi/Desk
top/pram/applied ai course/DonorsChoose_2018/train_data.csv')
resource_data = pd.read_csv('C:/Users/pramod reddy chandi/Des
ktop/pram/applied ai course/DonorsChoose_2018/resources.csv')
```

In [3]:

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

```
Number of data points in train data (109248, 1
7)
--------------------------------------------------
----
The attributes of data : ['Unnamed: 0' 'id' 't
eacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_c
ategory'
 'project_subject_categories' 'project_subject
_subcategories'
 'project_title' 'project_essay_1' 'project_es
say_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects
' 'project_is_approved']
```

In [4]:

```
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[4]:

	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

In [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]
```

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

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	sch
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

In [6]:

```
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[6]:

	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 [7]:

```
catogories = list(project_data['project_subject_categories'].
values)
# remove special characters from list of strings python: http
s://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-pyth
on/
# https://stackoverflow.com/questions/23669024/how-to-strip-a
-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whit
espace-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", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            j=j.replace('The','') # if we have the words "The
" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        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, inp
lace=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 [8]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# 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", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The " we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        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://stackoverf
low.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=l
ambda kv: kv[1]))
```

1.3 Text preprocessing

In [9]:

```
# 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 [10]:

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

Out[10]:

	Unnamed: 0	id	teach
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3
76127	37728	p043609	3f60494c61921b3b43ab61bdde29

In [11]:

1.4.2.3 Using Pretrained Models: TFIDF weighted W2V

In [12]:

```
# 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)
```

```
I have been fortunate enough to use the Fairy
Tale STEM kits in my classroom as well as the
STEM journals, which my students really enjoye
d.  I would love to implement more of the Lake
shore STEM kits in my classroom for the next s
chool year as they provide excellent and engag
ing STEM lessons.My students come from a varie
ty of backgrounds, including language and soci
oeconomic status.  Many of them don't have a l
ot of experience in science and engineering an
d these kits give me the materials to provide
these exciting opportunities for my students.E
ach month I try to do several science or STEM/
STEAM projects.  I would use the kits and robo
t to help guide my science instruction in enga
ging and meaningful ways.  I can adapt the kit
s to my current language arts pacing guide whe
re we already teach some of the material in th
e kits like tall tales (Paul Bunyan) or Johnny
 Appleseed.  The following units will be taugh
t in the next school year where I will impleme
```

nt these kits: magnets, motion, sink vs. float , robots. I often get to these units and don' t know If I am teaching the right way or using the right materials. The kits will give me additional ideas, strategies, and lessons to prepare my students in science.It is challengi ng to develop high quality science activities. These kits give me the materials I need to p rovide my students with science activities tha t will go along with the curriculum in my clas sroom. Although I have some things (like magn ets) in my classroom, I don't know how to use them effectively. The kits will provide me wi th the right amount of materials and show me h ow to use them in an appropriate way.

==========

I teach high school English to students with l earning and behavioral disabilities. My studen ts all vary in their ability level. However, t he ultimate goal is to increase all students l iteracy levels. This includes their reading, w riting, and communication levels.I teach a rea lly dynamic group of students. However, my stu dents face a lot of challenges. My students al l live in poverty and in a dangerous neighborh ood. Despite these challenges, I have students who have the the desire to defeat these chall enges. My students all have learning disabilit ies and currently all are performing below gra de level. My students are visual learners and will benefit from a classroom that fulfills th eir preferred learning style.The materials I a m requesting will allow my students to be prep ared for the classroom with the necessary supp lies. Too often I am challenged with students who come to school unprepared for class due t

o economic challenges. I want my students to be able to focus on learning and not how they will be able to get school supplies. The supplies will last all year. Students will be able to complete written assignments and maintain a classroom journal. The chart paper will be used to make learning more visual in class and to create posters to aid students in their learning. The students have access to a classroom printer. The toner will be used to print student work that is completed on the classroom Chromebooks.I want to try and remove all barriers for the students learning and create opportunities for learning. One of the biggest barriers is the students not having the resources to get pens, paper, and folders. My students will be able to increase their literacy skills because of this project.

==========

\"Life moves pretty fast. If you don't stop and look around once in awhile, you could miss it.\" from the movie, Ferris Bueller's Day Off. Think back...what do you remember about your grandparents? How amazing would it be to be able to flip through a book to see a day in their lives?My second graders are voracious readers! They love to read both fiction and nonfiction books. Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson. They also love to read about insects, space and plants. My students are hungry bookworms! My students are eager to learn and read about the world around them. My kids love to be at school and are like little sponges absorbing everything around them. Their parents work long hours and usually do not see t

heir children. My students are usually cared f
or by their grandparents or a family friend. M
ost of my students do not have someone who spe
aks English at home. Thus it is difficult for
my students to acquire language.Now think forw
ard... wouldn't it mean a lot to your kids, ni
eces or nephews or grandchildren, to be able t
o see a day in your life today 30 years from n
ow? Memories are so precious to us and being a
ble to share these memories with future genera
tions will be a rewarding experience. As part
of our social studies curriculum, students wi
ll be learning about changes over time. Stude
nts will be studying photos to learn about how
their community has changed over time. In pa
rticular, we will look at photos to study how
the land, buildings, clothing, and schools hav
e changed over time. As a culminating activit
y, my students will capture a slice of their h
istory and preserve it through scrap booking.
Key important events in their young lives will
be documented with the date, location, and na
mes. Students will be using photos from home
and from school to create their second grade
memories. Their scrap books will preserve th
eir unique stories for future generations to e
njoy.Your donation to this project will provid
e my second graders with an opportunity to lea
rn about social studies in a fun and creative
manner. Through their scrapbooks, children wi
ll share their story with others and have a hi
storical document for the rest of their lives.
==========
====
\"A person's a person, no matter how small.\"
(Dr.Seuss) I teach the smallest students with
the biggest enthusiasm for learning. My studen

ts learn in many different ways using all of o
ur senses and multiple intelligences. I use a
wide range of techniques to help all my studen
ts succeed. \r\nStudents in my class come from
 a variety of different backgrounds which make
s for wonderful sharing of experiences and cul
tures, including Native Americans.\r\nOur scho
ol is a caring community of successful learner
s which can be seen through collaborative stud
ent project based learning in and out of the c
lassroom. Kindergarteners in my class love to
work with hands-on materials and have many dif
ferent opportunities to practice a skill befor
e it is mastered. Having the social skills to
work cooperatively with friends is a crucial a
spect of the kindergarten curriculum.Montana i
s the perfect place to learn about agriculture
 and nutrition. My students love to role play
in our pretend kitchen in the early childhood
classroom. I have had several kids ask me, \"C
an we try cooking with REAL food?\" I will tak
e their idea and create \"Common Core Cooking
Lessons\" where we learn important math and wr
iting concepts while cooking delicious healthy
 food for snack time. My students will have a
grounded appreciation for the work that went i
nto making the food and knowledge of where the
 ingredients came from as well as how it's hea
lthy for their bodies. This project would expa
nd our learning of nutrition and agricultural
cooking recipes by having us peel our own appl
es to make homemade applesauce, make our own b
read, and mix up healthy plants from our class
room garden in the spring. We will also create
 our own cookbooks to be printed and shared wi
th families. \r\nStudents will gain math and l
iterature skills as well as a life long enjoym

ent for healthy cooking.nannan
==========
My classroom consists of twenty-two amazing si xth graders from different cultures and backgr ounds. They are a social bunch who enjoy worki ng in partners and working with groups. They a re hard-working and eager to head to middle sc hool next year. My job is to get them ready to make this transition and make it as smooth as possible. In order to do this, my students ne ed to come to school every day and feel safe a nd ready to learn. Because they are getting re ady to head to middle school, I give them lots of choice- choice on where to sit and work, t he order to complete assignments, choice of pr ojects, etc. Part of the students feeling safe is the ability for them to come into a welcom ing, encouraging environment. My room is color ful and the atmosphere is casual. I want them to take ownership of the classroom because we ALL share it together. Because my time with th em is limited, I want to ensure they get the m ost of this time and enjoy it to the best of t heir abilities.Currently, we have twenty-two d esks of differing sizes, yet the desks are sim ilar to the ones the students will use in midd le school. We also have a kidney table with cr ates for seating. I allow my students to choos e their own spots while they are working indep endently or in groups. More often than not, mo st of them move out of their desks and onto th e crates. Believe it or not, this has proven t o be more successful than making them stay at their desks! It is because of this that I am l ooking toward the “Flexible Seating” option fo r my classroom.\r\n The students look forward

```
to their work time so they can move around the
 room. I would like to get rid of the constric
ting desks and move toward more “fun” seating
options. I am requesting various seating so my
 students have more options to sit. Currently,
 I have a stool and a papasan chair I inherite
d from the previous sixth-grade teacher as wel
l as five milk crate seats I made, but I would
 like to give them more options and reduce the
 competition for the “good seats”. I am also r
equesting two rugs as not only more seating op
tions but to make the classroom more welcoming
 and appealing. In order for my students to be
 able to write and complete work without desks
, I am requesting a class set of clipboards. F
inally, due to curriculum that requires groups
 to work together, I am requesting tables that
 we can fold up when we are not using them to
leave more room for our flexible seating optio
ns.\r\nI know that with more seating options,
they will be that much more excited about comi
ng to school! Thank you for your support in ma
king my classroom one students will remember f
orever!nannan
================================================
====
```

In [13]:

```
# 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 [14]:

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

```
\"A person is a person, no matter how small.\"
 (Dr.Seuss) I teach the smallest students with
 the biggest enthusiasm for learning. My stude
nts learn in many different ways using all of
our senses and multiple intelligences. I use a
 wide range of techniques to help all my stude
nts succeed. \r\nStudents in my class come fro
m a variety of different backgrounds which mak
es for wonderful sharing of experiences and cu
ltures, including Native Americans.\r\nOur sch
ool is a caring community of successful learne
rs which can be seen through collaborative stu
dent project based learning in and out of the
classroom. Kindergarteners in my class love to
 work with hands-on materials and have many di
fferent opportunities to practice a skill befo
re it is mastered. Having the social skills to
 work cooperatively with friends is a crucial
aspect of the kindergarten curriculum.Montana
is the perfect place to learn about agricultur
e and nutrition. My students love to role play
```

```
 in our pretend kitchen in the early childhood
 classroom. I have had several kids ask me, \"
Can we try cooking with REAL food?\" I will ta
ke their idea and create \"Common Core Cooking
 Lessons\" where we learn important math and w
riting concepts while cooking delicious health
y food for snack time. My students will have a
 grounded appreciation for the work that went
into making the food and knowledge of where th
e ingredients came from as well as how it is h
ealthy for their bodies. This project would ex
pand our learning of nutrition and agricultura
l cooking recipes by having us peel our own ap
ples to make homemade applesauce, make our own
 bread, and mix up healthy plants from our cla
ssroom garden in the spring. We will also crea
te our own cookbooks to be printed and shared
with families. \r\nStudents will gain math and
 literature skills as well as a life long enjo
yment for healthy cooking.nannan
==========================================
====
```

In [15]:

```
# \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)
```

```
 A person is a person, no matter how small.  (
Dr.Seuss) I teach the smallest students with t
he biggest enthusiasm for learning. My student
s learn in many different ways using all of ou
r senses and multiple intelligences. I use a w
ide range of techniques to help all my student
```

s succeed. Students in my class come from a variety of different backgrounds which makes f or wonderful sharing of experiences and cultur es, including Native Americans. Our school is a caring community of successful learners whi ch can be seen through collaborative student p roject based learning in and out of the classr oom. Kindergarteners in my class love to work with hands-on materials and have many differen t opportunities to practice a skill before it is mastered. Having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the perfect place to learn about agriculture and nutrition. My students love to role play in ou r pretend kitchen in the early childhood class room. I have had several kids ask me, Can we try cooking with REAL food? I will take their idea and create Common Core Cooking Lessons where we learn important math and writing con cepts while cooking delicious healthy food for snack time. My students will have a grounded appreciation for the work that went into makin g the food and knowledge of where the ingredie nts came from as well as how it is healthy for their bodies. This project would expand our l earning of nutrition and agricultural cooking recipes by having us peel our own apples to ma ke homemade applesauce, make our own bread, an d mix up healthy plants from our classroom gar den in the spring. We will also create our own cookbooks to be printed and shared with famil ies. Students will gain math and literature skills as well as a life long enjoyment for he althy cooking.nannan

In [16]:

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

```
 A person is a person no matter how small Dr S
euss I teach the smallest students with the bi
ggest enthusiasm for learning My students lear
n in many different ways using all of our sens
es and multiple intelligences I use a wide ran
ge of techniques to help all my students succe
ed Students in my class come from a variety of
 different backgrounds which makes for wonderf
ul sharing of experiences and cultures includi
ng Native Americans Our school is a caring com
munity of successful learners which can be see
n through collaborative student project based
learning in and out of the classroom Kindergar
teners in my class love to work with hands on
materials and have many different opportunitie
s to practice a skill before it is mastered Ha
ving the social skills to work cooperatively w
ith friends is a crucial aspect of the kinderg
arten curriculum Montana is the perfect place
to learn about agriculture and nutrition My st
udents love to role play in our pretend kitche
n in the early childhood classroom I have had
several kids ask me Can we try cooking with RE
AL food I will take their idea and create Comm
on Core Cooking Lessons where we learn importa
nt math and writing concepts while cooking del
icious healthy food for snack time My students
 will have a grounded appreciation for the wor
k that went into making the food and knowledge
 of where the ingredients came from as well as
 how it is healthy for their bodies This proje
ct would expand our learning of nutrition and
```

```
agricultural cooking recipes by having us peel
 our own apples to make homemade applesauce ma
ke our own bread and mix up healthy plants fro
m our classroom garden in the spring We will a
lso create our own cookbooks to be printed and
 shared with families Students will gain math
and literature skills as well as a life long e
njoyment for healthy cooking nannan
```

In [17]:

```
# 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've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', '
yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "
it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', '
whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', '
being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', '
if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in
', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where',
 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', '
same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't",
'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "co
```

```
uldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", '
isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't",
'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
 \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:

```
# 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 stopw
ords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:58<00:00, 1
869.81it/s]
```

In [19]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[19]:

```
'a person person no matter small dr seuss i te
ach smallest students biggest enthusiasm learn
ing my students learn many different ways usin
g senses multiple intelligences i use wide ran
```

```
ge techniques help students succeed students c
lass come variety different backgrounds makes
wonderful sharing experiences cultures includi
ng native americans our school caring communit
y successful learners seen collaborative stude
nt project based learning classroom kindergart
eners class love work hands materials many dif
ferent opportunities practice skill mastered h
aving social skills work cooperatively friends
 crucial aspect kindergarten curriculum montan
a perfect place learn agriculture nutrition my
 students love role play pretend kitchen early
 childhood classroom i several kids ask can tr
y cooking real food i take idea create common
core cooking lessons learn important math writ
ing concepts cooking delicious healthy food sn
ack time my students grounded appreciation wor
k went making food knowledge ingredients came
well healthy bodies this project would expand
learning nutrition agricultural cooking recipe
s us peel apples make homemade applesauce make
 bread mix healthy plants classroom garden spr
ing we also create cookbooks printed shared fa
milies students gain math literature skills we
ll life long enjoyment healthy cooking nannan'
```

In [20]:

```
#Project essay word count

essay_word_count = []

for ess in project_data["essay"] :
    c = len(ess.split())
    essay_word_count.append(c)

project_data["essay_word_count"] = essay_word_count
```

In [21]:

```
project_data['preprocessed_essays'] = preprocessed_essays
```

In [22]:

```
import nltk
```

In [23]:

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

analyser = SentimentIntensityAnalyzer()

pos =[]
neg = []
neu = []
compound = []

for a in tqdm(project_data["preprocessed_essays"]) :
    b = analyser.polarity_scores(a)['neg']
    c = analyser.polarity_scores(a)['pos']
    d = analyser.polarity_scores(a)['neu']
    e = analyser.polarity_scores(a)['compound']
    neg.append(b)
    pos.append(c)
    neu.append(d)
    compound.append(e)
```

```
100%|██████████| 109248/109248 [12:50<00:00, 1
41.86it/s]
```

In [24]:

```
project_data["pos"] = pos
project_data["neg"] = neg
project_data["neu"] = neu
project_data["compound"] = compound
```

1.4 Preprocessing of $project_tit \leq$

In [25]:

```
# similarly you can preprocess the titles also
# similarly you can preprocess the titles also

project_data.columns
#sent1= decontracted(project_data['project_title'].values[200
00])
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e not in sto
pwords)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:02<00:00, 3
7409.78it/s]
```

In [26]:

```
#Project title word count
title_word_count = []

for a in project_data["project_title"] :
    b = len(a.split())
    title_word_count.append(b)
```

```
project_data["title_word_count"] = title_word_count
```

In [27]:

```
project_data['preprocessed_title'] = preprocessed_title
```

1.5 Preparing data for models

In [28]:

```
project_data.columns
```

Out[28]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
her_prefix', 'school_state',
       'Date', 'project_grade_category', 'proj
ect_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', '
project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'clean_categories', 'clean_subcategorie
s', 'essay', 'essay_word_count',
       'preprocessed_essays', 'pos', 'neg', 'n
eu', 'compound',
       'title_word_count', 'preprocessed_title
'],
      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 : nu
merical
    - price : numerical
```

In [29]:

```
Y=project_data['project_is_approved']
```

In [30]:

```
price_data = resource_data.groupby('id').agg({'price':'sum',
'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', ho
w='left')
```

In [31]:

```
column_values=['clean_categories', 'clean_subcategories', 'sc
hool_state', 'project_grade_category', 'teacher_prefix','prep
rocessed_essays','preprocessed_title' ,'price','quantity','te
acher_number_of_previously_posted_projects','pos','neg','neu',
'compound','title_word_count','essay_word_count']

def select_columns(dataframe, column_names):
    new_frame = dataframe.loc[:, column_names]
    return new_frame

process_columns=select_columns(project_data,column_values)
```

In [32]:

```
process_columns.head()
```

Out[32]:

	clean_categories	clean_subcategories	school_state	project_grade_category

0	Math_Science	AppliedSciences Health_LifeScience	CA	Grades PreK-2
1	SpecialNeeds	SpecialNeeds	UT	Grades 3-5
2	Literacy_Language	Literacy	CA	Grades PreK-2
3	AppliedLearning	EarlyDevelopment	GA	Grades PreK-2
4	Literacy_Language	Literacy	WA	Grades 3-5

In [33]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
from sklearn.model_selection import train_test_split

# X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, shuffle=Flase)# this is for time series split
X_train, X_test, y_train, y_test = train_test_split(process_columns, Y, test_size=0.33,random_state=42) # this is random splitting
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33 ,random_state=42) # this is random splitting


print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(49041, 16) (49041,)
```

```
(24155, 16) (24155,)
(36052, 16) (36052,)
==================================================
==================================================
========
```

In [34]:

```
print("train columns",X_train.columns)

print("cv columns",X_cv.columns)

print("test columns",X_test.columns)
```

```
train columns Index(['clean_categories', 'clea
n_subcategories', 'school_state',
       'project_grade_category', 'teacher_pref
ix', 'preprocessed_essays',
       'preprocessed_title', 'price', 'quantit
y',
       'teacher_number_of_previously_posted_pr
ojects', 'pos', 'neg', 'neu',
       'compound', 'title_word_count', 'essay_
word_count'],
      dtype='object')
cv columns Index(['clean_categories', 'clean_s
ubcategories', 'school_state',
       'project_grade_category', 'teacher_pref
ix', 'preprocessed_essays',
       'preprocessed_title', 'price', 'quantit
y',
       'teacher_number_of_previously_posted_pr
ojects', 'pos', 'neg', 'neu',
       'compound', 'title_word_count', 'essay_
word_count'],
      dtype='object')
test columns Index(['clean_categories', 'clean
_subcategories', 'school_state',
```

```
       'project_grade_category', 'teacher_pref
ix', 'preprocessed_essays',
       'preprocessed_title', 'price', 'quantit
y',
       'teacher_number_of_previously_posted_pr
ojects', 'pos', 'neg', 'neu',
       'compound', 'title_word_count', 'essay_
word_count'],
      dtype='object')
```

1.5.1 Vectorizing Categorical data

In [35]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_categories= CountVectorizer(vocabulary=list(sorted
_cat_dict.keys()), lowercase=False, binary=True)

vectorizer_categories.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer_categories.transform(X_
train['clean_categories'].values)
categories_one_hot_test = vectorizer_categories.transform(X_t
est['clean_categories'].values)
categories_one_hot_cv = vectorizer_categories.transform(X_cv[
'clean_categories'].values)

print(vectorizer_categories.get_feature_names())

print("Shape of train matrix after one hot encodig ",categori
es_one_hot_train.shape)
print("Shape of test matrix after one hot encodig ",categorie
s_one_hot_test.shape)
print("Shape of cv matrix after one hot encodig ",categories_
one_hot_cv.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'M
usic_Arts', 'AppliedLearning', 'SpecialNeeds',
 'Health_Sports', 'Math_Science', 'Literacy_La
nguage']
Shape of train matrix after one hot encodig  (
49041, 9)
Shape of test matrix after one hot encodig  (3
6052, 9)
Shape of cv matrix after one hot encodig  (241
55, 9)
```

In [36]:

```
# we use count vectorizer to convert the values into one
# splitting subcategories data
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_subcategories = CountVectorizer(vocabulary=list(so
rted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_subcategories.fit(X_train['clean_subcategories'].v
alues)

print(vectorizer_subcategories.get_feature_names())

sub_categories_one_hot_train = vectorizer_subcategories.trans
form(X_train['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer_subcategories.transf
orm(X_test['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer_subcategories.transfor
m(X_cv['clean_subcategories'].values)

print("Shape of train matrix after one hot encodig ",sub_cate
gories_one_hot_train.shape)
print("Shape of test matrix after one hot encodig ",sub_categ
ories_one_hot_test.shape)
print("Shape of cv matrix after one hot encodig ",sub_categor
ies_one_hot_cv.shape)
```

```
['Economics', 'CommunityService', 'FinancialLi
teracy', 'ParentInvolvement', 'Extracurricular
', 'Civics_Government', 'ForeignLanguages', 'N
utritionEducation', 'Warmth', 'Care_Hunger', '
SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_Car
eerPrep', 'Music', 'History_Geography', 'Healt
h_LifeScience', 'EarlyDevelopment', 'ESL', 'Gy
m_Fitness', 'EnvironmentalScience', 'VisualArt
s', 'Health_Wellness', 'AppliedSciences', 'Spe
cialNeeds', 'Literature_Writing', 'Mathematics
', 'Literacy']
Shape of train matrix after one hot encodig  (
49041, 30)
Shape of test matrix after one hot encodig  (3
6052, 30)
Shape of cv matrix after one hot encodig  (241
55, 30)
```

In [37]:

```
# we use count vectorizer to convert the values of categorica
l data :school_state
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_schoolstate= CountVectorizer()
vectorizer_schoolstate.fit(X_train['school_state'])

print(vectorizer_schoolstate.get_feature_names())

school_state_one_hot_train = vectorizer_schoolstate.transform
(X_train['school_state'].values)
school_state_one_hot_test = vectorizer_schoolstate.transform(
X_test['school_state'].values)
school_state_one_hot_cv = vectorizer_schoolstate.transform(X_
cv['school_state'].values)
```

```
print("Shape of train matrix after one hot encodig ",school_state_one_hot_train.shape)
print("Shape of test matrix after one hot encodig ",school_state_one_hot_test.shape)
print("Shape of cv matrix after one hot encodig ",school_state_one_hot_cv.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc
', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', '
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']
Shape of train matrix after one hot encodig  (
49041, 51)
Shape of test matrix after one hot encodig  (3
6052, 51)
Shape of cv matrix after one hot encodig  (241
55, 51)
```

In [38]:

```
#we use count vectorizer to convert the values of categorical
 data :project_grade_category
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_project_grade_category = CountVectorizer(stop_words=None)

k=X_train['project_grade_category']
l=X_test['project_grade_category']
m=X_test['project_grade_category']

k.replace(['Grades PreK-2', 'Grades 6-8', 'Grades 3-5','Grades 9-12'], ['A1', 'B2' ,'C3', 'D4'],inplace=True)
l.replace(['Grades PreK-2', 'Grades 6-8', 'Grades 3-5','Grades 9-12'], ['A1', 'B2' ,'C3', 'D4'],inplace=True)
m.replace(['Grades PreK-2', 'Grades 6-8', 'Grades 3-5','Grade
```

```
s 9-12'], ['A1', 'B2' ,'C3', 'D4'],inplace=True)

vectorizer_project_grade_category.fit(k)

project_grade_category_one_hot_train=vectorizer_project_grade
_category.transform(X_train['project_grade_category'].values)
project_grade_category_one_hot_test=vectorizer_project_grade_
category.transform(X_test['project_grade_category'].values)
project_grade_category_one_hot_cv=vectorizer_project_grade_ca
tegory.transform(X_cv['project_grade_category'].values)

print("Shape of train matrix after one hot encodig ",project_
grade_category_one_hot_train.shape)
print("Shape of test matrix after one hot encodig ",project_g
rade_category_one_hot_test.shape)
print("Shape of cv matrix after one hot encodig ",project_gra
de_category_one_hot_cv.shape)
```

```
Shape of train matrix after one hot encodig  (
49041, 4)
Shape of test matrix after one hot encodig  (3
6052, 4)
Shape of cv matrix after one hot encodig  (241
55, 4)
```

In [39]:

```
#we use count vectorizer to convert the values of categorical
 data : teacher_prefix
# getting error as we have null balues replacing them with 0
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_teacher_prefix = CountVectorizer()
project_data['teacher_prefix'].unique()

X_train['teacher_prefix'].fillna("", inplace = True)
X_test['teacher_prefix'].fillna("", inplace = True)
X_cv['teacher_prefix'].fillna("", inplace = True)
```

```
vectorizer_teacher_prefix.fit(X_train['teacher_prefix'].values
)
print(vectorizer_teacher_prefix.get_feature_names())

teacher_prefix_one_hot_train = vectorizer_teacher_prefix.tran
sform(X_train['teacher_prefix'].values)
teacher_prefix_one_hot_test = vectorizer_teacher_prefix.trans
form(X_test['teacher_prefix'].values)
teacher_prefix_one_hot_cv = vectorizer_teacher_prefix.transfo
rm(X_cv['teacher_prefix'].values)

print("Shape of train matrix after one hot encodig ",teacher_
prefix_one_hot_train.shape)
print("Shape of test matrix after one hot encodig ",teacher_p
refix_one_hot_test.shape)
print("Shape of cv matrix after one hot encodig ",teacher_pre
fix_one_hot_cv.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of train matrix after one hot encodig  (
49041, 5)
Shape of test matrix after one hot encodig  (3
6052, 5)
Shape of cv matrix after one hot encodig  (241
55, 5)
```

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

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [40]:

```
# We are considering only the words which appeared in at leas
t 10 documents(rows or projects).
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_bow_essay = CountVectorizer(min_df=10, ngram_range
 =(1,2),max_features=5000)
vectorizer_bow_essay.fit(X_train['preprocessed_essays'])

text_bow_train= vectorizer_bow_essay.transform(X_train['prepr
ocessed_essays'])
text_bow_test= vectorizer_bow_essay.transform(X_test['preproc
essed_essays'])
text_bow_cv= vectorizer_bow_essay.transform(X_cv['preprocesse
d_essays'])

print("Shape of train matrix after one hot encodig ",text_bow
_train.shape)
print("Shape of test matrix after one hot encodig ",text_bow_
test.shape)
print("Shape of cv matrix after one hot encodig ",text_bow_cv
.shape)
```

```
Shape of train matrix after one hot encodig  (
49041, 5000)
Shape of test matrix after one hot encodig  (3
6052, 5000)
Shape of cv matrix after one hot encodig  (241
55, 5000)
```

In [41]:

```
# before you vectorize the title make sure you preprocess it
from sklearn.feature_extraction.text import CountVectorizer
```

```
vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit(X_train['preprocessed_title'])

title_bow_train = vectorizer_bow_title.transform(X_train['pre
processed_title'])
title_bow_test = vectorizer_bow_title.transform(X_test['prepr
ocessed_title'])
title_bow_cv= vectorizer_bow_title.transform(X_cv['preprocess
ed_title'])

print("Shape of train matrix after one hot encodig title_bow"
,title_bow_train.shape)
print("Shape of test matrix after one hot encodig title_bow",
title_bow_test.shape)
print("Shape of cv matrix after one hot encodig title_bow",ti
tle_bow_cv.shape)
```

```
Shape of train matrix after one hot encodig ti
tle_bow (49041, 91)
Shape of test matrix after one hot encodig tit
le_bow (36052, 91)
Shape of cv matrix after one hot encodig title
_bow (24155, 91)
```

1.5.2.2 TFIDF vectorizer

In [42]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay= TfidfVectorizer(min_df=10,ngram_range
 =(1,2),max_features=7500)
vectorizer_tfidf_essay.fit(X_train['preprocessed_essays'])

text_tfidf_train= vectorizer_tfidf_essay.transform(X_train['p
```

```
reprocessed_essays'])
text_tfidf_test= vectorizer_tfidf_essay.transform(X_test['pre
processed_essays'])
text_tfidf_cv = vectorizer_tfidf_essay.transform(X_cv['prepro
cessed_essays'])

print("Shape of train matrix after one hot encodig ",text_tfi
df_train.shape)
print("Shape of test matrix after one hot encodig ",text_tfid
f_test.shape)
print("Shape of cv matrix after one hot encodig ",text_tfidf_
cv.shape)
```

```
Shape of train matrix after one hot encodig  (
49041, 7500)
Shape of test matrix after one hot encodig  (3
6052, 7500)
Shape of cv matrix after one hot encodig  (241
55, 7500)
```

In [43]:

```
# Similarly you can vectorize for title also

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(X_train['preprocessed_title'])

title_tfidf_train = vectorizer_tfidf_title.transform(X_train[
'preprocessed_title'])
title_tfidf_test = vectorizer_tfidf_title.transform(X_test['p
reprocessed_title'])
title_tfidf_cv = vectorizer_tfidf_title.transform(X_cv['prepr
ocessed_title'])

print("Shape of train matrix after one hot encodig ",title_tf
idf_train.shape)
```

```
print("Shape of test matrix after one hot encodig ",title_tfidf_test.shape)
print("Shape of cv matrix after one hot encodig ",title_tfidf_cv.shape)
```

```
Shape of train matrix after one hot encodig  (49041, 91)
Shape of test matrix after one hot encodig  (36052, 91)
Shape of cv matrix after one hot encodig  (24155, 91)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [44]:

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
```

In [45]:

```
i=0
list_of_sentance_train=[]
for sentance in X_train['preprocessed_essays']:
    list_of_sentance_train.append(sentance.split())
```

In [46]:

```
# this line of code trains your w2v model on the give list of sentances
w2v_model=Word2Vec(list_of_sentance_train,min_count=25,size=50, workers=32)
```

In [47]:

```
w2v_words = list(w2v_model.wv.vocab)
```

```
print("number of words that occured minimum 25 times ",len(w2
v_words))
print("sample words ", w2v_words[0:50])
```

```
number of words that occured minimum 25 times
 8649
sample words  ['we', 'low', 'income', 'school'
, 'texas', 'near', 'dallas', 'fort', 'my', 'st
udents', 'always', 'apply', 'best', 'effort',
'due', 'beyond', 'control', 'lack', 'additiona
l', 'funding', 'purchase', 'supplies', 'classr
oom', 'our', 'desires', 'produce', '21st', 'ce
ntury', 'learners', 'sometimes', 'need', 'help
', 'even', 'though', 'brightest', 'enthusiasti
c', 'around', 'they', 'eager', 'learn', 'want'
, 'provide', 'would', 'use', 'books', 'daily',
 'reinforce', 'lessons', 'american', 'revoluti
on']
```

In [48]:

```
# average Word2Vec of essays
# compute average word2vec for each review.
essay_vectors_train = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sent in tqdm(list_of_sentance_train): # for each review/s
entence
    sent_vec = np.zeros(50) # as word vectors are of zero len
gth 50, you might need to change this to 300 if you use googl
e's w2v
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
```

```
        sent_vec /= cnt_words
    essay_vectors_train.append(sent_vec)
essay_vectors_train = np.array(essay_vectors_train)
print(essay_vectors_train.shape)
print(essay_vectors_train[0])
```

```
100%|██████████| 49041/49041 [02:29<00:00, 329
.12it/s]
```

```
(49041, 50)
[-0.71928713 -0.47792962  0.1036844  -0.787324
99 -0.57420449  0.25555406
 -0.00236374 -0.49353458  0.30305412  0.372054
74  0.92357103 -0.39046017
 -0.1246915   0.08349687  0.1700186   1.083002
4   0.22677555 -0.02780715
  0.0197929   0.49607713 -0.0286198   0.261104
16 -0.25146864 -0.46226857
  0.30343399  0.70015849 -0.34209903  0.019242
58 -0.14962942  0.35760852
  0.21681893  0.44508216  0.15862708 -0.692285
48 -0.35769462 -0.56733366
  0.3640078  -0.24497056 -0.4789511   0.277177
85  0.63389754 -0.34260041
 -0.61697854  0.10391176 -0.0406382   0.022911
13 -0.45079223  0.22106107
  0.42738568 -0.19192756]
```

In [49]:

```
i=0
list_of_sentance_cv=[]
for sentance in X_cv['preprocessed_essays']:
    list_of_sentance_cv.append(sentance.split())
```

In [50]:

```
# average Word2Vec
```

```
# compute average word2vec for each review.
essay_vectors_cv = []; # the avg-w2v for each sentence/review
 is stored in this list
for sent in tqdm(list_of_sentance_cv): # for each review/sent
ence
    sent_vec = np.zeros(50) # as word vectors are of zero len
gth 50, you might need to change this to 300 if you use googl
e's w2v
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    essay_vectors_cv.append(sent_vec)
essay_vectors_cv = np.array(essay_vectors_cv)
print(essay_vectors_cv.shape)
print(essay_vectors_cv[0])
```

```
100%|██████████| 24155/24155 [01:10<00:00, 341
.09it/s]
```

```
(24155, 50)
[-0.49510997 -0.91730499  0.26328437 -0.353577
59 -0.34404254  0.25423687
 -0.26256536 -0.78362508  0.25586309 -0.001392
65  0.5216866  -0.5882191
 -0.00436863 -0.27422285  0.51356697  1.232692
69  0.18512844 -0.11802516
 -0.90730044  1.03605692 -0.35188645  0.429878
48 -0.27969142 -0.35380829
  0.39801197  0.58135989 -0.33026655  0.039600
85 -0.38267427  0.49312856
  0.20182547  0.53807878  0.4538499  -0.874156
05 -0.24724144 -0.41314018
```

```
  0.08983388 -0.62727061 -0.22310478  0.440146
51  0.38644395 -0.51137581
 -1.04415616  0.33170003 -0.36342517  0.378231
14 -0.36379444 -0.60758678
 -0.04546012  0.080174  ]
```

In [51]:

```
i=0
list_of_sentance_test=[]
for sentance in X_test['preprocessed_essays']:
    list_of_sentance_test.append(sentance.split())
```

In [52]:

```
# average Word2Vec
# compute average word2vec for each review.
essay_vectors_test = []; # the avg-w2v for each sentence/revi
ew is stored in this list
for sent in tqdm(list_of_sentance_test): # for each review/se
ntence
    sent_vec = np.zeros(50) # as word vectors are of zero len
gth 50, you might need to change this to 300 if you use googl
e's w2v
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    essay_vectors_test.append(sent_vec)
essay_vectors_test = np.array(essay_vectors_test)
print(essay_vectors_test.shape)
print(essay_vectors_test[0])
```

```
100%|██████████| 36052/36052 [01:45<00:00, 343
.16it/s]
```

```
(36052, 50)
[-0.78736916 -0.51413786  0.31149084 -0.540816
95 -0.67424056  0.34677591
  0.20171977 -0.46601164  0.33795609  0.342500
04  0.63774066 -0.19810071
  0.1565317  -0.39926447  0.24747765  0.925747
11 -0.20682891 -0.08680184
  0.18597113  0.7008063   0.07486268  0.072722
35 -0.33449357 -0.60230249
 -0.14206684  1.01864593 -0.01606888 -0.307380
75 -0.25070406 -0.17618433
 -0.05739199 -0.08970815  0.41108211 -0.331490
15  0.05131065 -0.57380467
 -0.04109803 -0.90058931 -0.19531263 -0.122906
03  0.50597863  0.20290311
 -0.47192344 -0.02600262  0.15270138  0.151380
02 -0.10764534  0.14417334
  0.45710415 -0.30143113]
```

In [53]:

```
#similarly doing it for preprocessed title
i=0
list_of_sentance_train=[]
for sentance in X_train['preprocessed_title']:
    list_of_sentance_train.append(sentance.split())
```

In [54]:

```
# this line of code trains your w2v model on the give list of
 sentances
w2v_model=Word2Vec(list_of_sentance_train,min_count=5,size=50
, workers=16)
```

In [55]:

```
w2v_words = list(w2v_model.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v
_words))
print("sample words ", w2v_words[0:50])
```

```
number of words that occured minimum 5 times
92
sample words  ['my', 'first', 'graders', 'eage
r', 'learn', 'world', 'around', 'they', 'come'
, 'school', 'day', 'full', 'enthusiasm', 'genu
inely', 'love', 'learning', 'our', 'diverse',
'class', 'includes', 'students', 'variety', 'c
ultural', 'economic', 'backgrounds', 'many', '
homes', 'parents', 'not', 'afford', 'simply',
'know', 'importance', 'books', 'important', 'p
rovide', 'environment', 'rich', 'literature',
'reading', 'i', 'want', 'lifelong', 'learners'
, 'best', 'way', 'used', 'magazines', 'past',
'kids']
```

In [56]:

```
# compute average word2vec for each review.
title_vectors_train = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sent in tqdm(list_of_sentance_train): # for each review/s
entence
    sent_vec = np.zeros(50) # as word vectors are of zero len
gth 50, you might need to change this to 300 if you use googl
e's w2v
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
```

```
    if cnt_words != 0:
        sent_vec /= cnt_words
    title_vectors_train.append(sent_vec)
title_vectors_train = np.array(title_vectors_train)
print(title_vectors_train.shape)
print(title_vectors_train[0])
```

```
100%|██████████| 49041/49041 [00:23<00:00, 211
9.30it/s]
```

```
(49041, 50)
[ 0.09420543 -0.20772289 -0.03672475  0.037062
1   0.08333692 -0.08049148
 -0.22191689 -0.05652462  0.23295018 -0.049019
36  0.17364979 -0.19951483
 -0.10793917 -0.17337105 -0.21158673  0.058058
99 -0.29645236 -0.02384486
  0.03955337 -0.06159221  0.12663943 -0.093937
58 -0.13148467 -0.17604463
 -0.04345671  0.20790434 -0.22029195 -0.164829
56  0.33567881  0.23413967
 -0.18043016  0.07862208 -0.06888222 -0.025487
05 -0.07271733  0.048897
 -0.34239116  0.20014654 -0.16977077  0.138508
14  0.0649581  -0.11028406
 -0.00655713 -0.20136692 -0.039062   -0.203405
78  0.33310974 -0.09640513
  0.11076641  0.11348761]
```

In [57]:

```
i=0
list_of_sentance_cv=[]
for sentance in X_cv['preprocessed_title']:
    list_of_sentance_cv.append(sentance.split())

# compute average word2vec for each review.
title_vectors_cv = []; # the avg-w2v for each sentence/review
```

```
 is stored in this list
for sent in tqdm(list_of_sentance_cv): # for each review/sent
ence
    sent_vec = np.zeros(50) # as word vectors are of zero len
gth 50, you might need to change this to 300 if you use googl
e's w2v
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    title_vectors_cv.append(sent_vec)
title_vectors_cv = np.array(title_vectors_cv)
print(title_vectors_cv.shape)
print(title_vectors_cv[0])
```

```
100%|██████████| 24155/24155 [00:11<00:00, 217
0.96it/s]

(24155, 50)
[ 0.09420543 -0.20772289 -0.03672475  0.037062
1   0.08333692 -0.08049148
 -0.22191689 -0.05652462  0.23295018 -0.049019
36  0.17364979 -0.19951483
 -0.10793917 -0.17337105 -0.21158673  0.058058
99 -0.29645236 -0.02384486
  0.03955337 -0.06159221  0.12663943 -0.093937
58 -0.13148467 -0.17604463
 -0.04345671  0.20790434 -0.22029195 -0.164829
56  0.33567881  0.23413967
 -0.18043016  0.07862208 -0.06888222 -0.025487
05 -0.07271733  0.048897
 -0.34239116  0.20014654 -0.16977077  0.138508
14  0.0649581  -0.11028406
```

```
 -0.00655713 -0.20136692 -0.039062   -0.203405
78  0.33310974 -0.09640513
  0.11076641  0.11348761]
```

In [58]:

```
i=0
list_of_sentance_test=[]
for sentance in X_test['preprocessed_title']:
    list_of_sentance_test.append(sentance.split())

# compute average word2vec for each review.
title_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sent in tqdm(list_of_sentance_test): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, you might need to change this to 300 if you use google's w2v
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            vec = w2v_model.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    title_vectors_test.append(sent_vec)
title_vectors_test = np.array(title_vectors_test)
print(title_vectors_test.shape)
print(title_vectors_test[0])
```

```
100%|██████████| 36052/36052 [00:16<00:00, 2123.06it/s]
```

```
(36052, 50)
[ 0.09420543 -0.20772289 -0.03672475  0.037062
```

```
1   0.08333692 -0.08049148
 -0.22191689 -0.05652462  0.23295018 -0.049019
36  0.17364979 -0.19951483
 -0.10793917 -0.17337105 -0.21158673  0.058058
99 -0.29645236 -0.02384486
  0.03955337 -0.06159221  0.12663943 -0.093937
58 -0.13148467 -0.17604463
 -0.04345671  0.20790434 -0.22029195 -0.164829
56  0.33567881  0.23413967
 -0.18043016  0.07862208 -0.06888222 -0.025487
05 -0.07271733  0.048897
 -0.34239116  0.20014654 -0.16977077  0.138508
14  0.0649581  -0.11028406
 -0.00655713 -0.20136692 -0.039062   -0.203405
78  0.33310974 -0.09640513
  0.11076641  0.11348761]
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [60]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['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(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [61]:

```
# stronging variables into pickle files python: http://www.je
ssicayung.com/how-to-use-pickle-to-save-and-load-variables-in
-python/
# make sure you have the glove_vectors file
```

```
with open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai course/DonorsChoose_2018/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [62]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['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((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            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_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))
```

```
100%|██████████| 49041/49041 [01:36<00:00, 509
```

```
.74it/s]
```

```
49041
300
```

In [63]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['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((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            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_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
100%|██████████| 36052/36052 [01:11<00:00, 504
```

```
.26it/s]
```

```
36052
300
```

In [64]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['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((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            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_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

```
100%|██████████| 24155/24155 [00:49<00:00, 488
```

```
.55it/s]
```

```
24155
300
```

In [65]:

```
# Similarly you can vectorize for title also
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_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(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [66]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_title_train = []; # the avg-w2v for each sentence/r
eview is stored in this list
for sentence in tqdm(X_train['preprocessed_title']): # for ea
ch review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
```

```
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title_train.append(vector)

print(len(tfidf_w2v_title_train))
print(len(tfidf_w2v_title_train[0]))
```

```
100%|██████████| 49041/49041 [01:01<00:00, 791.10it/s]
```

```
49041
300
```

In [67]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_title_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_title']): # 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((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
```

```
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title_test.append(vector)

print(len(tfidf_w2v_title_test))
print(len(tfidf_w2v_title_test[0]))
```

```
100%|██████████| 36052/36052 [00:46<00:00, 781.72it/s]
```

```
36052
300
```

In [68]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_title_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_title']): # 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((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
```

```
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title_cv.append(vector)

print(len(tfidf_w2v_title_cv))
print(len(tfidf_w2v_title_cv[0]))
```

```
100%|██████████| 24155/24155 [00:32<00:00, 748.91it/s]
```

```
24155
300
```

1.5.3 Vectorizing Numerical features

In [69]:

```
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 [70]:

```
#scaling of price feature

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

# 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.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = Normalizer()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # fin
ding the mean and standard deviation of this data


# Now standardize the data with above maen and variance.
price_standardized_train= price_scalar.transform(X_train['pri
ce'].values.reshape(-1, 1))
price_standardized_test= price_scalar.transform(X_test['price
'].values.reshape(-1, 1))
price_standardized_cv= price_scalar.transform(X_cv['price'].v
alues.reshape(-1, 1))

print("After vectorizations")
print(price_standardized_train.shape, y_train.shape)
print(price_standardized_test.shape, y_test.shape)
print(price_standardized_cv.shape, y_cv.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(36052, 1) (36052,)
(24155, 1) (24155,)
```

In [71]:

```
#scaling of qunatity feature

# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4
&t=530s
# standardization sklearn: https://scikit-learn.org/stable/mo
dules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import Normalizer
```

```
# 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.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

quantity_scalar = Normalizer()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1))
 # finding the mean and standard deviation of this data


# Now standardize the data with above maen and variance.
quantity_standardized_train= quantity_scalar.transform(X_trai
n['quantity'].values.reshape(-1, 1))
quantity_standardized_test= quantity_scalar.transform(X_test[
'quantity'].values.reshape(-1, 1))
quantity_standardized_cv= quantity_scalar.transform(X_cv['qua
ntity'].values.reshape(-1, 1))

print("After vectorizations")
print(quantity_standardized_train.shape, y_train.shape)
print(quantity_standardized_test.shape, y_test.shape)
print(quantity_standardized_cv.shape, y_cv.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(36052, 1) (36052,)
(24155, 1) (24155,)
```

In [72]:

```
#scaling of teachers number of previously posted projects

from sklearn.preprocessing import Normalizer

normalizer_projects_num = Normalizer()
```

```
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer_projects_num.fit(X_train['teacher_number_of_previo
usly_posted_projects'].values.reshape(-1,1))

prev_projects_train = normalizer_projects_num.transform(X_tra
in['teacher_number_of_previously_posted_projects'].values.res
hape(-1,1))
prev_projects_cv = normalizer_projects_num.transform(X_cv['te
acher_number_of_previously_posted_projects'].values.reshape(-1
,1))
prev_projects_test = normalizer_projects_num.transform(X_test
['teacher_number_of_previously_posted_projects'].values.resha
pe(-1,1))

print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [73]:

```
# normalixing the title word count

from sklearn.preprocessing import Normalizer
```

```
normalizer_title_word = Normalizer()

normalizer_title_word.fit(X_train['title_word_count'].values.
reshape(-1,1))

title_word_count_train = normalizer_title_word.transform(X_tr
ain['title_word_count'].values.reshape(-1,1))
title_word_count_cv = normalizer_title_word.transform(X_cv['t
itle_word_count'].values.reshape(-1,1))
title_word_count_test = normalizer_title_word.transform(X_tes
t['title_word_count'].values.reshape(-1,1))

print("After vectorizations")
print(title_word_count_train.shape, y_train.shape)
print(title_word_count_cv.shape, y_cv.shape)
print(title_word_count_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
==================================================
==================================================
========
```

In [74]:

```
# normalixing the essay word count

from sklearn.preprocessing import Normalizer

normalizer_ess_count = Normalizer()

normalizer_ess_count.fit(X_train['essay_word_count'].values.r
eshape(-1,1))

essay_word_count_train = normalizer_ess_count.transform(X_tra
```

```
in['essay_word_count'].values.reshape(-1,1))
essay_word_count_cv = normalizer_ess_count.transform(X_cv['es
say_word_count'].values.reshape(-1,1))
essay_word_count_test = normalizer_ess_count.transform(X_test
['essay_word_count'].values.reshape(-1,1))

print("After vectorizations")
print(essay_word_count_train.shape, y_train.shape)
print(essay_word_count_cv.shape, y_cv.shape)
print(essay_word_count_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [75]:

```
#normalizing the data for  essay sentiment-pos
from sklearn.preprocessing import Normalizer
normalizer_pos = Normalizer()

normalizer_pos.fit(X_train['pos'].values.reshape(-1,1))

essay_sent_pos_train = normalizer_pos.transform(X_train['pos'
].values.reshape(-1,1))
essay_sent_pos_cv = normalizer_pos.transform(X_cv['pos'].valu
es.reshape(-1,1))
essay_sent_pos_test = normalizer_pos.transform(X_test['pos'].
values.reshape(-1,1))

print("After vectorizations")
print(essay_sent_pos_train.shape, y_train.shape)
print(essay_sent_pos_cv.shape, y_cv.shape)
print(essay_sent_pos_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
```

```
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [76]:

```
#normalizing the data for  essay sentiment-neg
from sklearn.preprocessing import Normalizer

normalizer_neg= Normalizer()

normalizer_neg.fit(X_train['neg'].values.reshape(-1,1))

essay_sent_neg_train = normalizer_neg.transform(X_train['neg'
].values.reshape(-1,1))
essay_sent_neg_cv = normalizer_neg.transform(X_cv['neg'].valu
es.reshape(-1,1))
essay_sent_neg_test = normalizer_neg.transform(X_test['neg'].
values.reshape(-1,1))

print("After vectorizations")
print(essay_sent_neg_train.shape, y_train.shape)
print(essay_sent_neg_cv.shape, y_cv.shape)
print(essay_sent_neg_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [77]:

```
#normalizing the data for  essay sentiment-neu
from sklearn.preprocessing import Normalizer

normalizer_nue= Normalizer()

normalizer_nue.fit(X_train['neu'].values.reshape(-1,1))
```

```
essay_sent_nue_train = normalizer_nue.transform(X_train['neu'
].values.reshape(-1,1))
essay_sent_nue_cv = normalizer_nue.transform(X_cv['neu'].valu
es.reshape(-1,1))
essay_sent_nue_test = normalizer_nue.transform(X_test['neu'].
values.reshape(-1,1))

print("After vectorizations")
print(essay_sent_nue_train.shape, y_train.shape)
print(essay_sent_nue_cv.shape, y_cv.shape)
print(essay_sent_nue_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [78]:

```
#normalizing the data for  essay sentiment-compound
from sklearn.preprocessing import Normalizer

normalizer_compound= Normalizer()

normalizer_compound.fit(X_train['compound'].values.reshape(-1,
1))

essay_sent_comp_train = normalizer_compound.transform(X_train
['compound'].values.reshape(-1,1))
essay_sent_comp_cv = normalizer_compound.transform(X_cv['comp
ound'].values.reshape(-1,1))
essay_sent_comp_test = normalizer_compound.transform(X_test['
compound'].values.reshape(-1,1))

print("After vectorizations")
print(essay_sent_comp_train.shape, y_train.shape)
print(essay_sent_comp_cv.shape, y_cv.shape)
print(essay_sent_comp_test.shape, y_test.shape)
```

```
print("="*100)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
==================================================
==================================================
========
```

1.5.4 Merging all the above features

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

In [79]:

```
from scipy.sparse import hstack

#define categorical and numerical features
cat_num_train=hstack((school_state_one_hot_train,categories_one_hot_train,sub_categories_one_hot_train,teacher_prefix_one_hot_train,project_grade_category_one_hot_train,price_standardized_train, quantity_standardized_train, prev_projects_train,
 title_word_count_train, essay_word_count_train, essay_sent_pos_train, essay_sent_neg_train, essay_sent_nue_train, essay_sent_comp_train))

cat_num_test=hstack((school_state_one_hot_test,categories_one_hot_test,sub_categories_one_hot_test,teacher_prefix_one_hot_test,project_grade_category_one_hot_test, price_standardized_test, quantity_standardized_test, prev_projects_test, title_word_count_test, essay_word_count_test, essay_sent_pos_test, essay_sent_neg_test, essay_sent_nue_test, essay_sent_comp_test
```

```
))

cat_num_cv=hstack((school_state_one_hot_cv,categories_one_hot
_cv,sub_categories_one_hot_cv,teacher_prefix_one_hot_cv,proje
ct_grade_category_one_hot_cv, price_standardized_cv, quantity
_standardized_cv, prev_projects_cv, title_word_count_cv, essa
y_word_count_cv, essay_sent_pos_cv, essay_sent_neg_cv, essay_
sent_nue_cv, essay_sent_comp_cv))

#combining categorical  numerical ,project_title(BOW)  and pr
eprocessed_essay (BOW)
set1_train = hstack((cat_num_train, text_bow_train,title_bow_
train))
set1_test = hstack((cat_num_test, text_bow_test,title_bow_tes
t))
set1_cv = hstack((cat_num_cv, text_bow_cv,title_bow_cv))

#categorical +numerical + project_title(TFIDF)+ preprocessed_
essay (TFIDF)
set2_train = hstack((cat_num_train, text_tfidf_train, title_t
fidf_train))
set2_test = hstack((cat_num_test, text_tfidf_test, title_tfid
f_test))
set2_cv = hstack((cat_num_cv, text_tfidf_cv, title_tfidf_cv))

#categorical ,numerical + project_title(AVG W2V)+ preprocesse
d_essay (AVG W2V)

set3_train = hstack((cat_num_train, essay_vectors_train,title
_vectors_train))
set3_test = hstack((cat_num_test, essay_vectors_test,title_ve
ctors_test))
set3_cv = hstack((cat_num_cv, essay_vectors_cv,title_vectors_
cv))

#categorical ,numerical+project_title(TFIDF W2V)+ preprocesse
d_essay (TFIDF W2V)
```

```
set4_train = hstack((cat_num_train, tfidf_w2v_vectors_train,
tfidf_w2v_title_train))
set4_test = hstack((cat_num_test, tfidf_w2v_vectors_test, tfi
df_w2v_title_test))
set4_cv = hstack((cat_num_cv, tfidf_w2v_vectors_cv, tfidf_w2v
_title_cv))
```

In [80]:

```
#saving all the variables for future use

import pickle
f=open('set1_svm.pckl','wb')
pickle.dump([set1_train, set1_test, set1_cv],f)
f.close()
```

In [81]:

```
import pickle
f=open('set2_svm.pckl','wb')
pickle.dump([set2_train, set2_test, set2_cv],f)
f.close()
```

In [82]:

```
import pickle
f=open('set3_svm.pckl','wb')
pickle.dump([set3_train, set3_test, set3_cv],f)
f.close()
```

In [83]:

```
import pickle
f=open('set4_svm.pckl','wb')
pickle.dump([set4_train, set4_test, set4_cv],f)
f.close()
```

In [84]:

```
import pickle
f=open('svm_y_values.pckl','wb')
pickle.dump([y_train,y_test,y_cv],f)
f.close()
```

In [85]:

```
import pickle
f=open('cat_num.pckl','wb')
pickle.dump([cat_num_train, cat_num_test, cat_num_cv],f)
f.close()
```

In [86]:

```
import pickle
f=open('text_train.pckl','wb')
pickle.dump([text_tfidf_train, text_tfidf_test, text_tfidf_cv
],f)
f.close()
```

In [81]:

```
'''f=open('num_cat.pckl','rb')
set_train1,set_test1,set_cv1=pickle.load(f)
f.close()'''
```

In [10]:

```
import pickle as pickle
#with open('C:/Users/pramod reddy chandi/Desktop/pram/applied
 ai course/DonorsChoose_2018/cat_num.pckl', 'rb') as f:
f=open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai
course/DonorsChoose_2018/cat_num.pckl','rb')
cat_num_train, cat_num_test, cat_num_cv=pickle.load(f)
f.close()
```


```
FileNotFoundError                         Trac
eback (most recent call last)
<ipython-input-10-35db17a09e6b> in <module>
      1 import pickle as pickle
      2 #with open('C:/Users/pramod reddy chan
di/Desktop/pram/applied ai course/DonorsChoose
_2018/cat_num.pckl', 'rb') as f:
----> 3 f=open('C:/Users/pramod reddy chandi/D
esktop/pram/applied ai course/DonorsChoose_201
8/cat_num.pckl','rb')
      4 cat_num_train, cat_num_test, cat_num_c
v=pickle.load(f)
      5 f.close()

FileNotFoundError: [Errno 2] No such file or d
irectory: 'C:/Users/pramod reddy chandi/Deskto
p/pram/applied ai course/DonorsChoose_2018/cat
_num.pckl'
```

In [2]:

```
cat_num_train.shape
```

Out[2]:

```
(49041, 108)
```

In [3]:

```
type(cat_num_train)
```

Out[3]:

```
scipy.sparse.coo.coo_matrix
```

In [4]:

```
import pickle as pickle
#with open('C:/Users/pramod reddy chandi/Desktop/pram/applied
 ai course/DonorsChoose_2018/cat_num.pckl', 'rb') as f:
```

```
f=open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai course/DonorsChoose_2018/set2.pckl','rb')
set2_train, set2_test, set2_cv=pickle.load(f)
f.close()
```

In [5]:

```
set2t=set2_train.tocsr()
text_tfidf_train=set2t[:,108:7608]
print(text_tfidf_train.shape)
```

```
(49041, 7500)
```

In [6]:

```
set2t_test=set2_test.tocsr()
text_tfidf_test=set2t_test[:,108:7608]
print(text_tfidf_test.shape)
```

```
(36052, 7500)
```

In [7]:

```
set2t_cv=set2_cv.tocsr()
text_tfidf_cv=set2t_cv[:,108:7608]
print(text_tfidf_cv.shape)
```

```
(24155, 7500)
```

Assignment 7: SVM

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

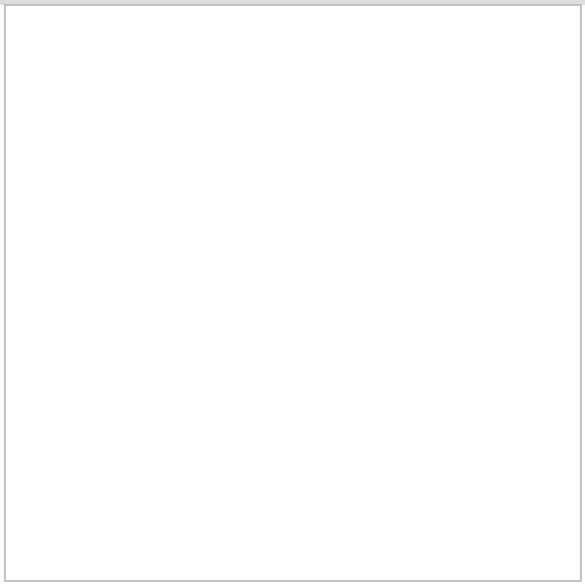
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: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

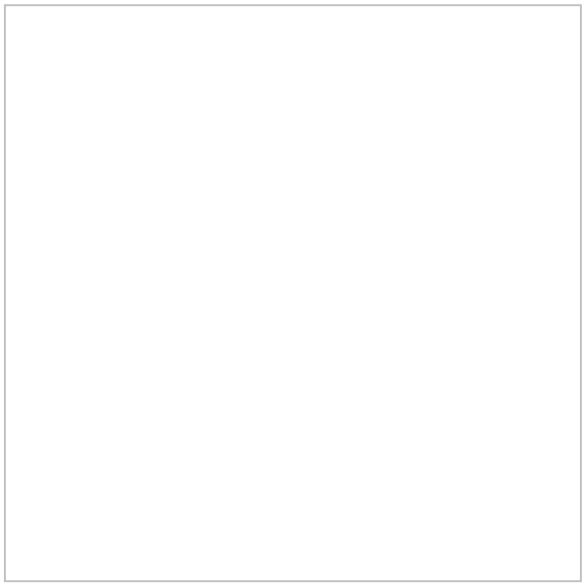
    - Find the best hyper parameter which will give the maximum AUC value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Representation of results**

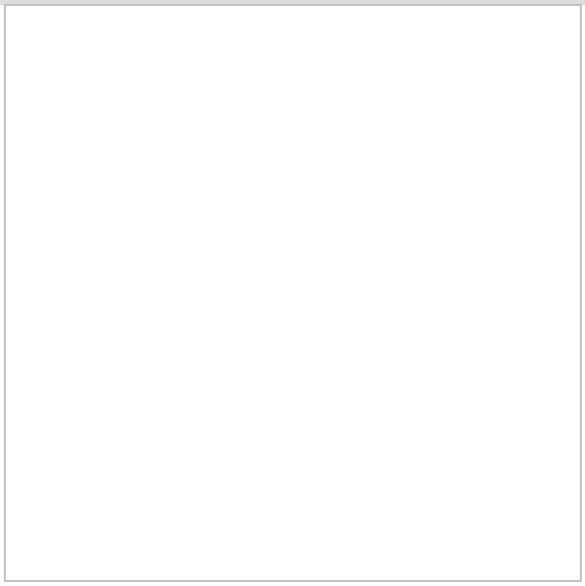
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure.



- 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 with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

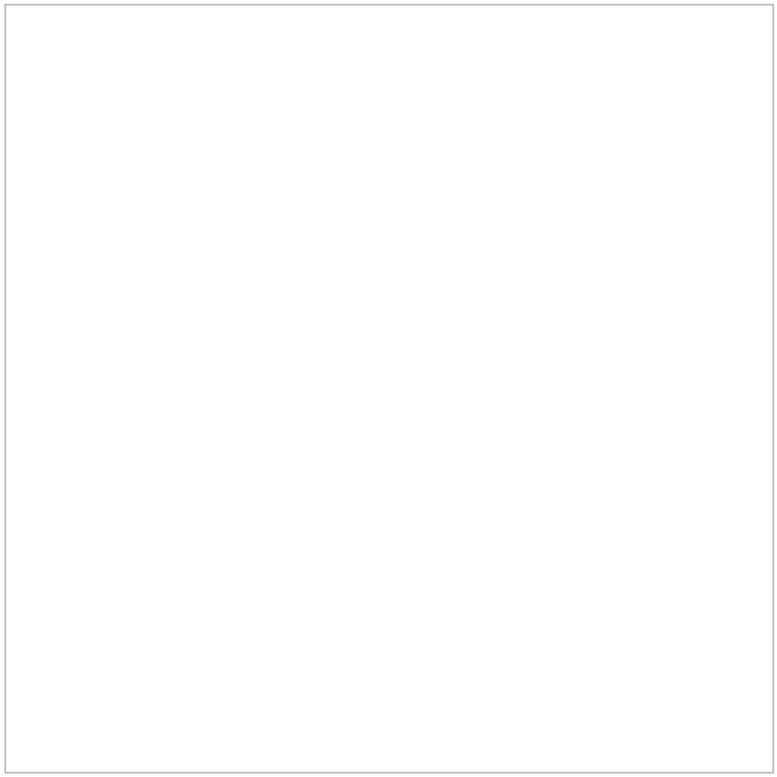


4. **[Task-2] Apply the Support Vector Machines on these features by finding the best hyper paramter as suggested in step 2 and step 3**

    - 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 on TfidfVectorizer of essay text, choose the number of components (`n_components`) using elbow method** : 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



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.

2. Support Vector Machines

Support Vector Machines on set 1

In [87]:

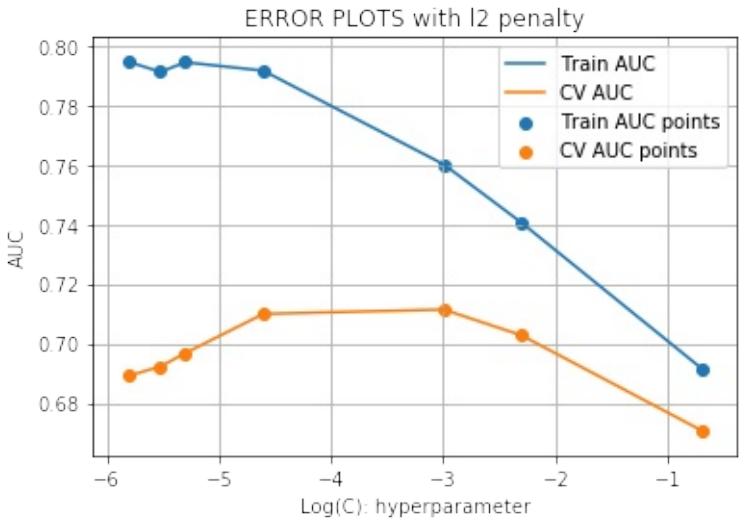
```
#Since we considered SGD classifier for linearsvm we need to
consider both hyperparameters L1 and L2 penalty
#and also hyperparemeter alpha of SGD is which is inverse of
C and no of points
##alpha=1/(C*m)
#C hypeparameter is regularization hyperparameter

# variables ready

X_tr=set1_train.tocsr()
X_cr=set1_cv.tocsr()
X_te=set1_test.tocsr()
```

Set1:doing SGD classification with L2 penalty

In [88]:

```
%%time

#doing Logistic regression on L2 penalty
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005,0.004,0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha= i,penalty='l
2',random_state=42,class_weight='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')
```

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l2 penalty")
plt.grid()
plt.show()
```




```
Wall time: 6.93 s
```

In [100]:

```
# We could see that the best hyperparameter for log(C) is -3
import math
k_best=math.pow(2.718281,-3)
```

In [101]:

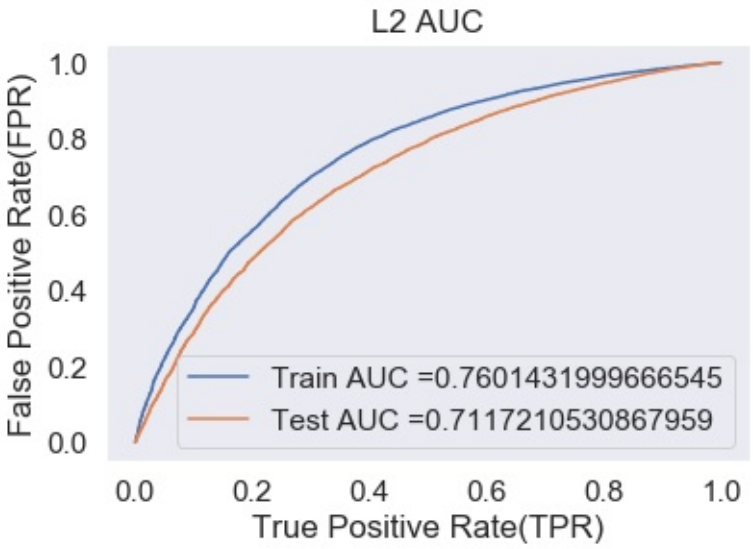
```
k_best
```

Out[101]:

```
0.0497871138891618
```

In [102]:

```
# finding AUC for train and test for L2 penalty
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import SGDClassifier

model = SGDClassifier(loss='hinge',alpha = k_best,penalty='l2
',random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L2 AUC")
plt.grid()
plt.show()
```


L2 AUC
False Positive Rate(FPR)
1.0
0.8
0.6
0.4
0.2
0.0
Train AUC =0.7601431999666545
Test AUC =0.7117210530867959
0.0
0.2
0.4
0.6
0.8
1.0
True Positive Rate(TPR)

Confusion matrix

In [103]:

```
def predict(proba, threshould, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [104]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.422
[[ 3752  3752]
 [ 6025 35512]]
```

In [105]:

```
conf_matr_df_trainl2_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

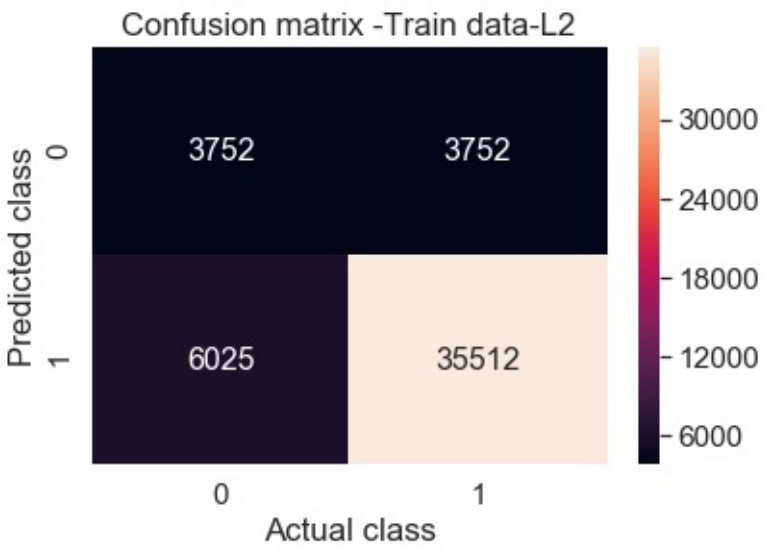
```
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.422
```

In [106]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl2_1, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -Train data-L2")
```

Out[106]:

```
Text(0.5, 1.0, 'Confusion matrix -Train data-L
2')
```



In [107]:

```
from sklearn.metrics import confusion_matrix
```

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.134
[[ 3207  2220]
 [ 8423 22202]]
```

In [108]:

```
conf_matr_df_testl2_1 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

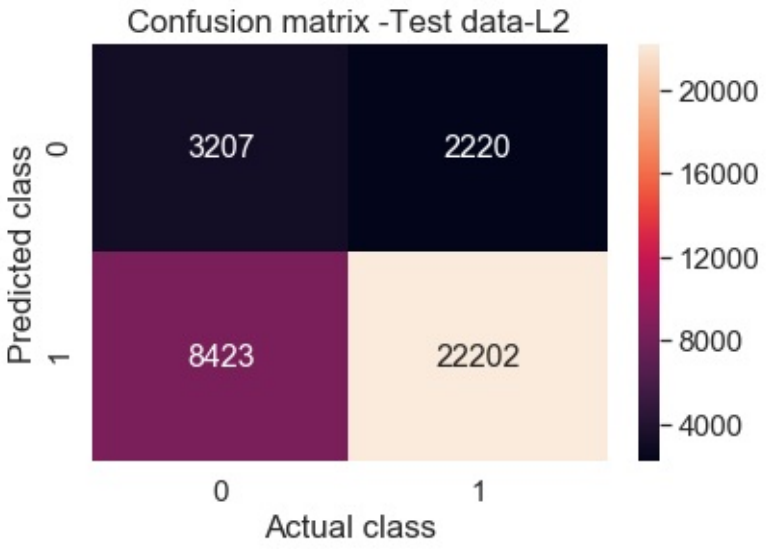
```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.134
```

In [109]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_testl2_1, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -Test data-L2")
```

Out[109]:

```
Text(0.5, 1.0, 'Confusion matrix -Test data-L2
')
```

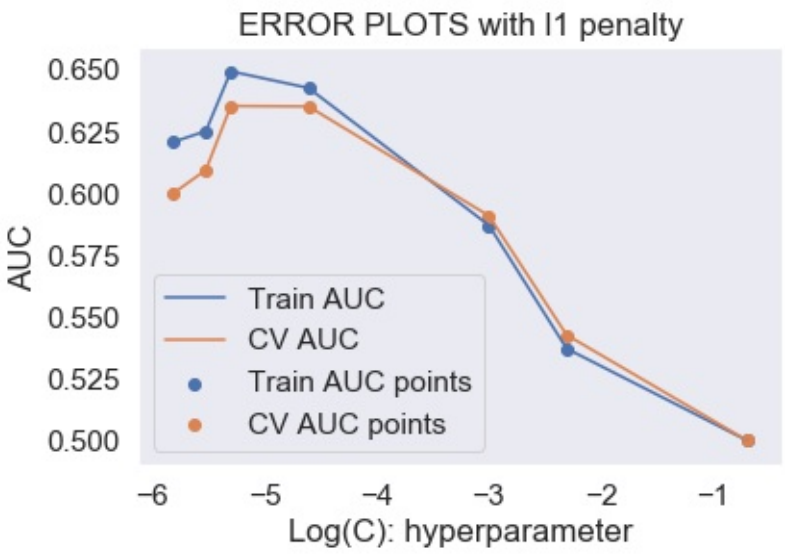

Confusion matrix -Test data-L2
3207
2220
8423
22202
Predicted class
0
1
0
1
Actual class
20000
16000
12000
8000
4000

Set1:doing SGD classification with L1 penalty

In [110]:

```
%%time
#doing Logistic regression on L1 penalty

import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import math
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes
]
Target scores, can either be probability estimates of the pos
itive class, confidence values, or non-thresholded measure of
decisions (as returned by “decision_function” on some classif
iers).
For binary y_true, y_score is supposed to be the score of the
 class with greater label.
"""
train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005, 0.004, 0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha= i,penalty='l
1',random_state=42,class_weight='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
```

```
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l1 penalty")
plt.grid()
plt.show()
```




```
Wall time: 3.2 s
```

In [111]:

```
# We could see that the best hyperparameter for log(C) is -4.
5 for l1 penalty
import math
k_best=math.pow(2.718281,-4.5)
```

In [112]:

```
k_best
```

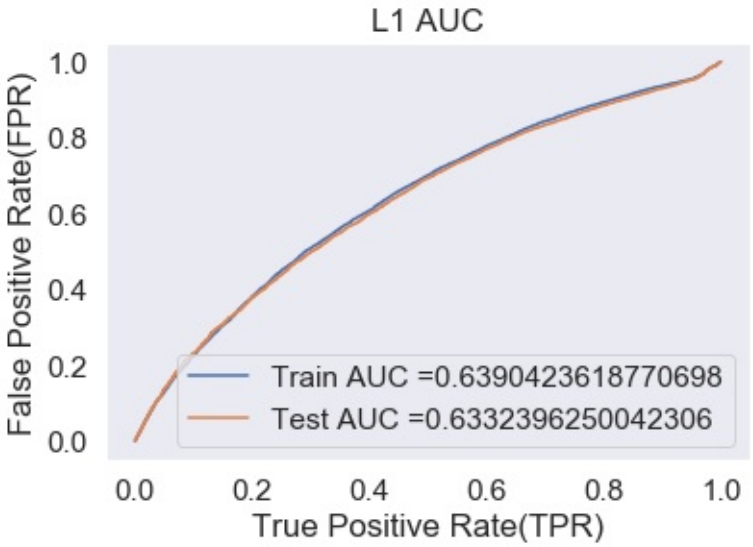
Out[112]:

```
0.011109011774007511
```

In [113]:

```
# finding AUC for train and test for L1 penalty

from sklearn.metrics import roc_curve, auc

model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l1'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
```

```
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L1 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [114]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.24
[[ 3752  3752]
```

```
 [12470 29067]]
```

In [115]:

```
conf_matr_df_trainl1_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

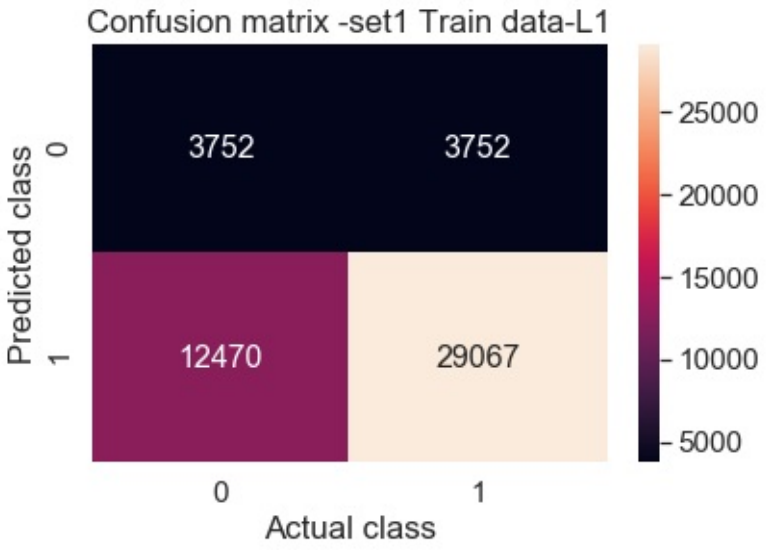
```
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.24
```

In [116]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl1_1, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set1 Train data-L1")
```

Out[116]:

```
Text(0.5, 1.0, 'Confusion matrix -set1 Train d
ata-L1')
```



In [117]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.001
[[ 3564  1863]
 [13970 16655]]
```

In [118]:

```
conf_matr_df_testl1_1 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.001
```

In [119]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_testl1_1, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 1:Test data-L1")
```

Out[119]:

```
Text(0.5, 1.0, 'Confusion matrix -set 1:Test d
ata-L1')
```


Confusion matrix -set 1:Test data-L1
Predicted class
0
1
3564
1863
13970
16655
0
1
Actual class
15000
12500
10000
7500
5000
2500

Support Vector Machines on set 2

In [120]:

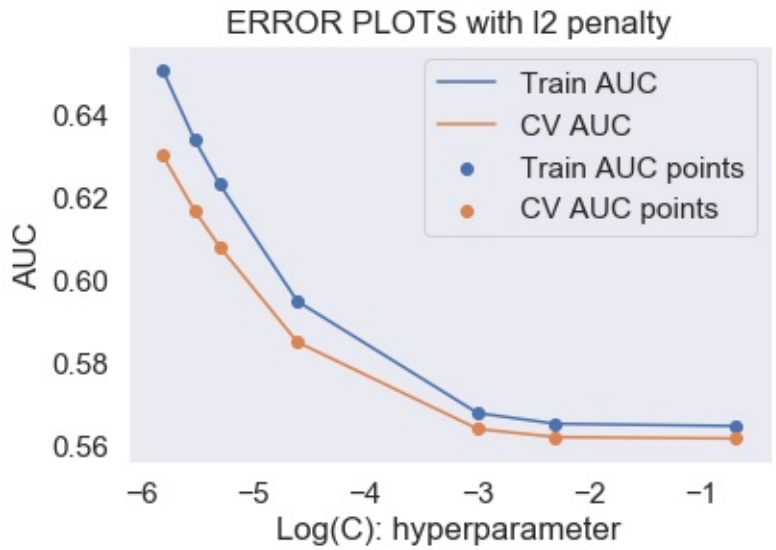
```
#Since we considered SGD classifier for linearsvm we need to
consider both hyperparameters L1 and L2 penalty
#and also hyperparemeter alpha of SGD is which is inverse of
C and no of points
##alpha=1/(C*m)
#C hypeparameter is regularization hyperparameter
# variables ready

X_tr=set2_train.tocsr()
X_cr=set2_cv.tocsr()
X_te=set2_test.tocsr()
```

Set2:doing Logistic regression with L2 penalty

In [121]:

```
%%time

#doing Logistic regression on L2 penalty
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005,0.004,0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha = i,penalty='
l2',random_state=42,class_weight='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')
```

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l2 penalty")
plt.grid()
plt.show()
```




```
Wall time: 2.33 s
```

In [122]:

```
# We could see that the best hyperparameter for log(C) is -6
import math
k_best=math.pow(2.718281,-6)
```

In [123]:

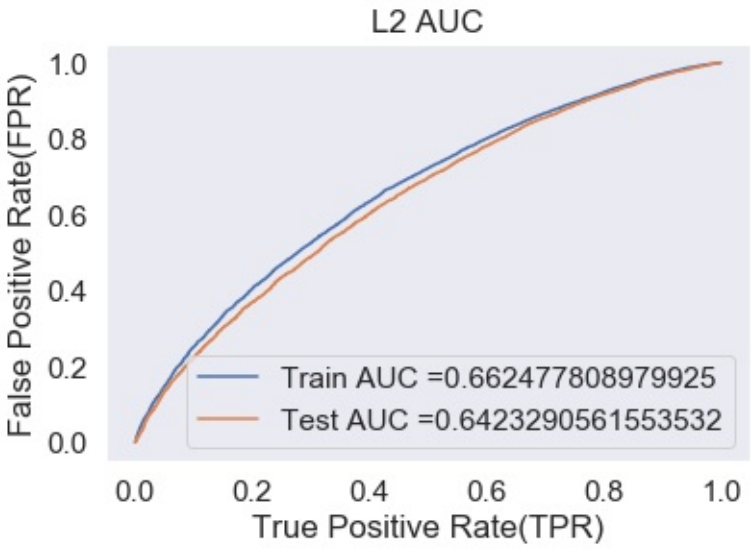
```
k_best
```

Out[123]:

0.0024787567094123678

In [124]:

```
# finding AUC for train and test for L2 penalty
from sklearn.metrics import roc_curve, auc


model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l2'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L2 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [125]:

```
def predict(proba, threshould, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [126]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999998224
117004 for threshold -0.324
[[ 3751  3753]
 [11566 29971]]
```

In [127]:

```
conf_matr_df_trainl2_2 = pd.DataFrame(confusion_matrix(y_trai
n, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)
), range(2),range(2))
```

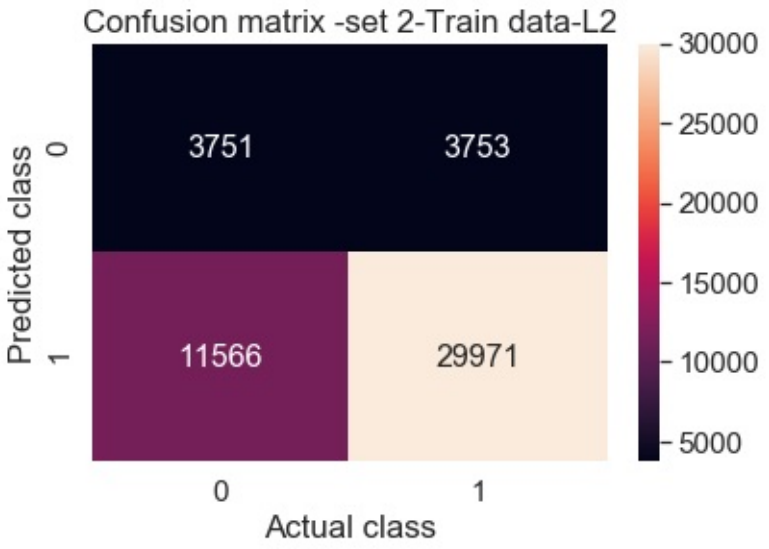
```
the maximum value of tpr*(1-fpr) 0.24999998224
117004 for threshold -0.324
```

In [128]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl2_2, annot=True,annot_kws={"si
ze": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 2-Train data-L2")
```

Out[128]:

```
Text(0.5, 1.0, 'Confusion matrix -set 2-Train
data-L2')
```



In [129]:

```
from sklearn.metrics import confusion_matrix
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.103
[[ 3402  2025]
 [13031 17594]]
```

In [130]:

```
conf_matr_df_testl2_2 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.103
```

In [131]:

```
sns.set(font_scale=1.4)#for label size
```

```
sns.heatmap(conf_matr_df_testl2_2, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 2-Test data-L2")
```

Out[131]:

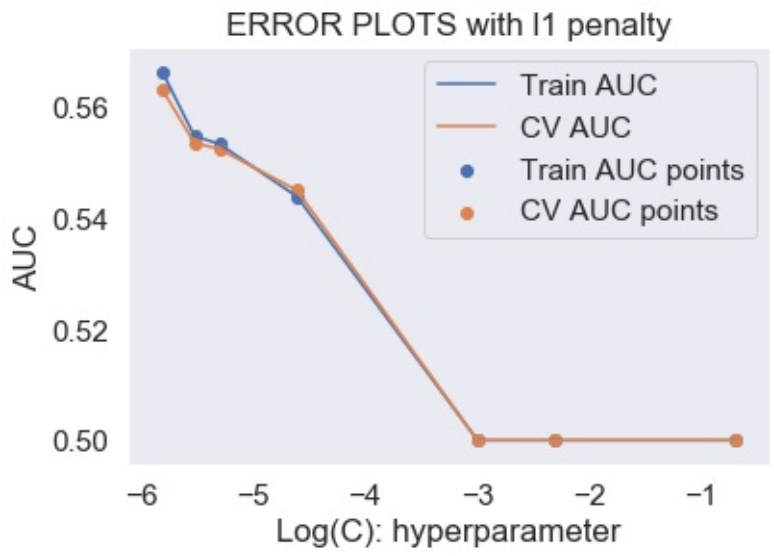
```
Text(0.5, 1.0, 'Confusion matrix -set 2-Test d
ata-L2')
```

Set2:doing SGD classification with L1 penalty

In [132]:

```
%%time
#doing Logistic regression on L1 penalty

import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import math
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes
]
Target scores, can either be probability estimates of the pos
itive class, confidence values, or non-thresholded measure of
decisions (as returned by “decision_function” on some classif
iers).
For binary y_true, y_score is supposed to be the score of the
 class with greater label.
"""
train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005, 0.004, 0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha= i,penalty='l
1',random_state=42,class_weight='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
```

```
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l1 penalty")
plt.grid()
plt.show()
```




```
Wall time: 3.04 s
```

In [133]:

```
# We could see that the best hyperparameter for log(C) is -6
import math
k_best=math.pow(2.718281,-6)
```

In [134]:

```
k_best
```

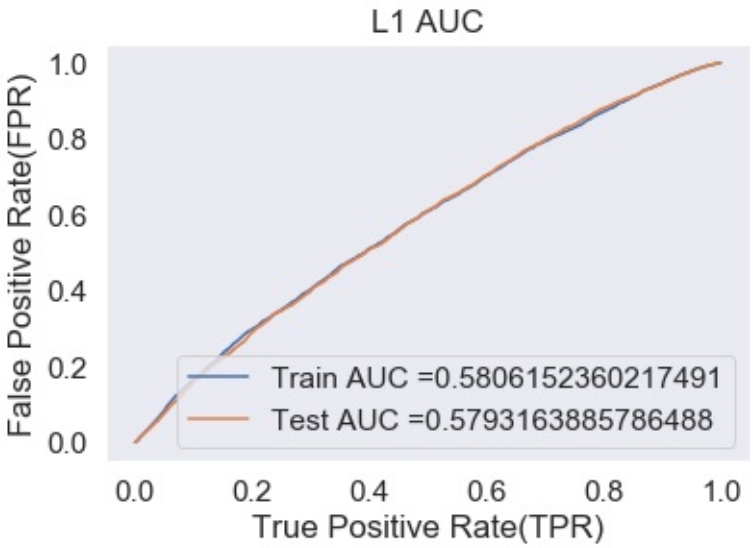
Out[134]:

```
0.0024787567094123678
```

In [135]:

```
# finding AUC for train and test for L2 penalty
from sklearn.metrics import roc_curve, auc


model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l1'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
```

```
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L1 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [136]:

```
def predict(proba, threshould, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
```

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

In [137]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.277
[[ 3752  3752]
 [16213 25324]]
```

In [138]:

```
conf_matr_df_trainl1_2 = pd.DataFrame(confusion_matrix(y_trai
n, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)
), range(2),range(2))
```

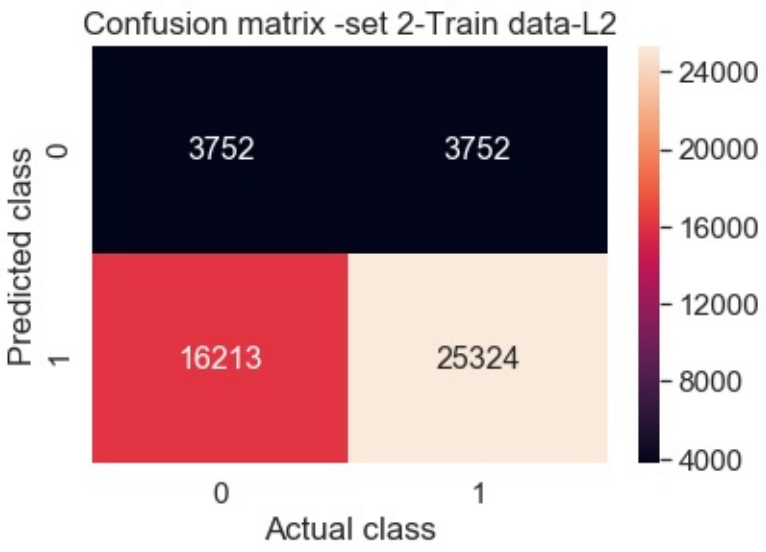
```
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.277
```

In [139]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl1_2, annot=True,annot_kws={"si
ze": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 2-Train data-L2")
```

Out[139]:

```
Text(0.5, 1.0, 'Confusion matrix -set 2-Train
data-L2')
```



In [140]:

```
from sklearn.metrics import confusion_matrix
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.025
[[ 3457  1970]
 [16217 14408]]
```

In [141]:

```
conf_matr_df_testl1_2 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.025
```

In [142]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_testl1_2, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 2-Test data-L2")
```

Out[142]:

```
Text(0.5, 1.0, 'Confusion matrix -set 2-Test data-L2')
```

Support Vector Machines on set 3

In [143]:

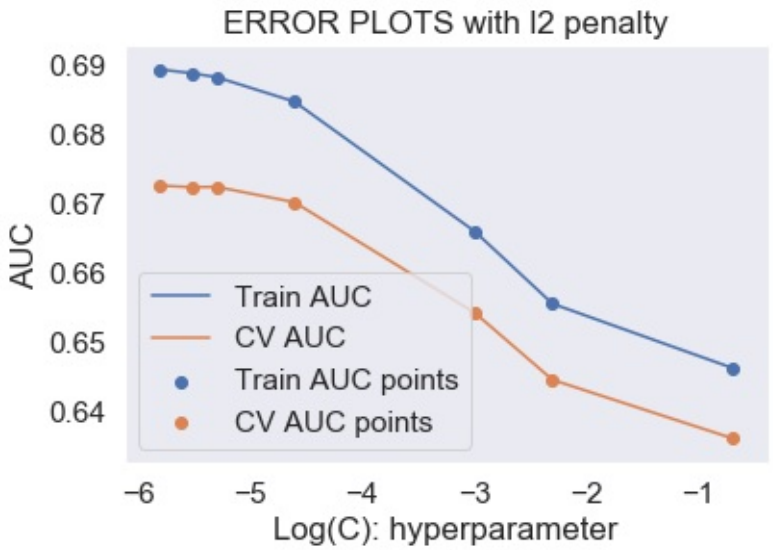
```
#Since we considered SGD classifier for linearsvm we need to
consider both hyperparameters L1 and L2 penalty
#and also hyperparemeter alpha of SGD is which is inverse of
C and no of points
##alpha=1/(C*m)
#C hypeparameter is regularization hyperparameter
# variables ready

X_tr=set3_train.tocsr()
X_cr=set3_cv.tocsr()
X_te=set3_test.tocsr()
```

Set3:doing Logistic regression with L2 penalty

In [144]:

```
%%time

#doing Logistic regression on L2 penalty
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005,0.004,0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha = i,penalty='
l2',class_weight='balanced',random_state=42,n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')
```

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l2 penalty")
plt.grid()
plt.show()
```




```
Wall time: 1.25 s
```

In [145]:

```
# We could see that the best hyperparameter for log(C) is -5.
5
import math
k_best=math.pow(2.718281,-5.5)
```

In [146]:

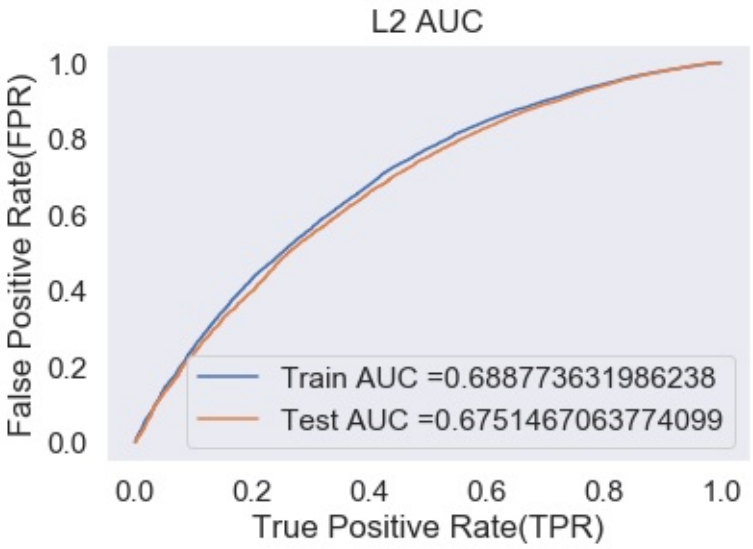
```
k_best
```

Out[146]:

0.004086778288928742

In [147]:

```
# finding AUC for train and test for L2 penalty
from sklearn.metrics import roc_curve, auc


model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l2'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L2 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [148]:

```
def predict(proba, threshould, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [149]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999998224
117004 for threshold -0.398
[[ 3751  3753]
 [ 9393 32144]]
```

In [150]:

```
conf_matr_df_trainl2_3 = pd.DataFrame(confusion_matrix(y_trai
n, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)
), range(2),range(2))
```

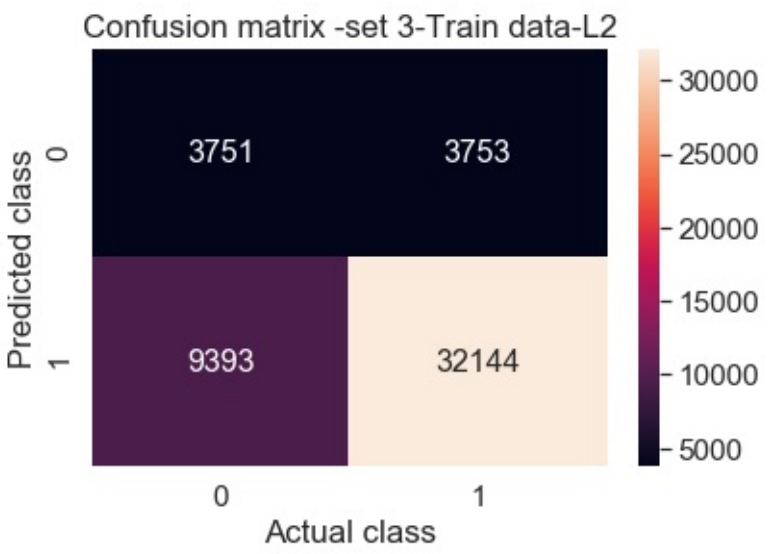
```
the maximum value of tpr*(1-fpr) 0.24999998224
117004 for threshold -0.398
```

In [151]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl2_3, annot=True,annot_kws={"si
ze": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 3-Train data-L2")
```

Out[151]:

```
Text(0.5, 1.0, 'Confusion matrix -set 3-Train
data-L2')
```



In [152]:

```
from sklearn.metrics import confusion_matrix
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
17069 for threshold -0.068
[[ 3458  1969]
 [11713 18912]]
```

In [153]:

```
conf_matr_df_testl2_3 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999151
17069 for threshold -0.068
```

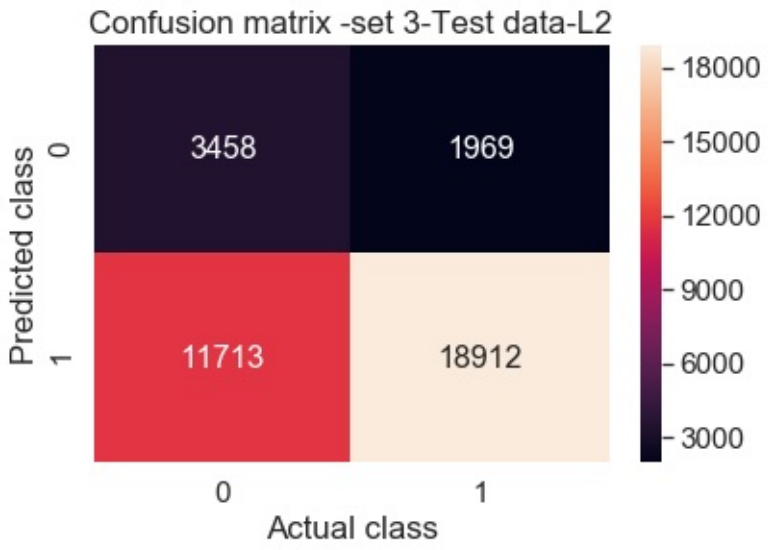
In [154]:

```
sns.set(font_scale=1.4)#for label size
```

```
sns.heatmap(conf_matr_df_testl2_3, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 3-Test data-L2")
```

Out[154]:

```
Text(0.5, 1.0, 'Confusion matrix -set 3-Test d
ata-L2')
```

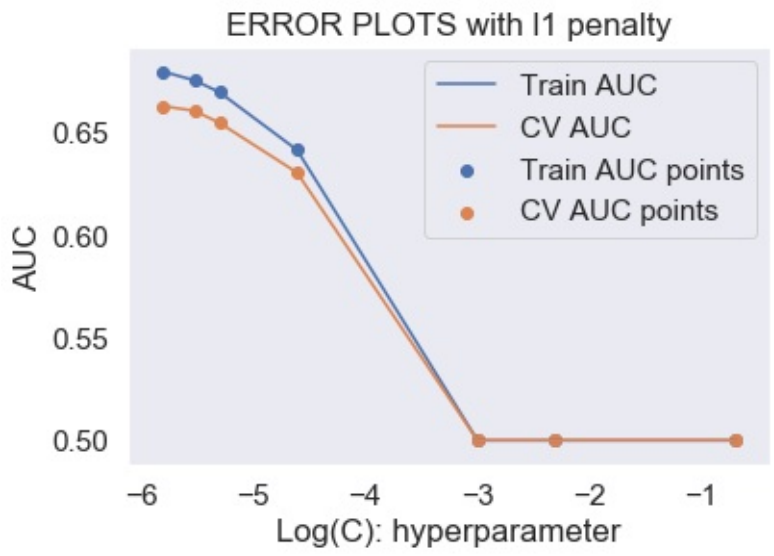


Set3:doing SGD classification with L1 penalty

In [155]:

```
%%time
#doing Logistic regression on L1 penalty

import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import math
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes
]
Target scores, can either be probability estimates of the pos
itive class, confidence values, or non-thresholded measure of
decisions (as returned by “decision_function” on some classif
iers).
For binary y_true, y_score is supposed to be the score of the
 class with greater label.
"""
train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005, 0.004, 0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha= i,penalty='l
1',class_weight='balanced',random_state=42,n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
```

```
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l1 penalty")
plt.grid()
plt.show()
```




```
Wall time: 1.95 s
```

In [156]:

```
# We could see that the best hyperparameter for log(C) is -6
 for l1 penalty
import math
k_best=math.pow(2.718281,-6)
```

In [157]:

```
k_best
```

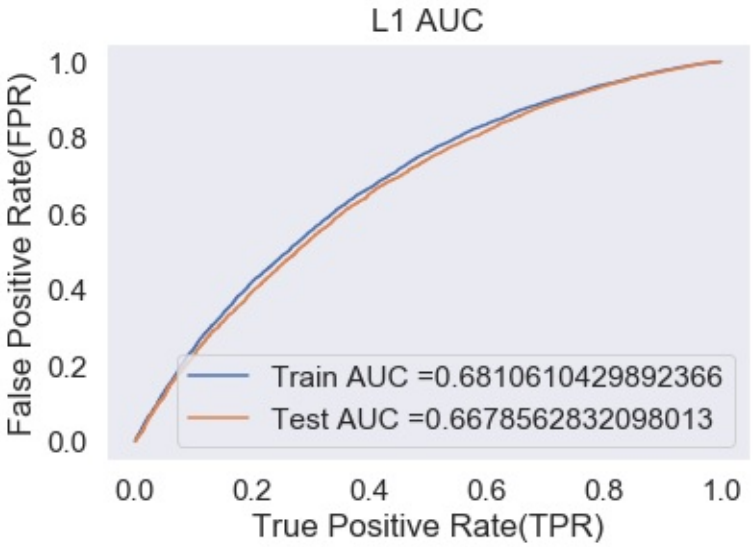
Out[157]:

```
0.0024787567094123678
```

In [158]:

```
# finding AUC for train and test for L1 penalty

from sklearn.metrics import roc_curve, auc

model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l1'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
```

```
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L1 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [159]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999998224
117007 for threshold -0.375
[[ 3753  3751]
```

```
 [ 9904 31633]]
```

In [160]:

```
conf_matr_df_trainl1_3 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

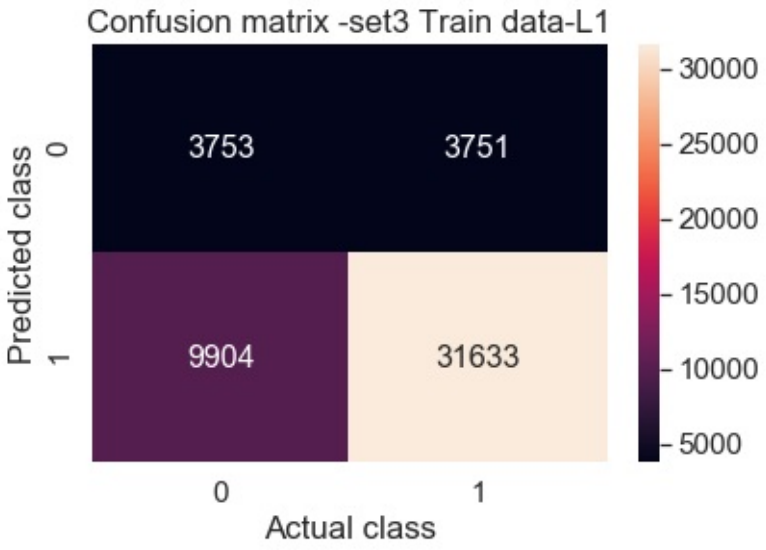
```
the maximum value of tpr*(1-fpr) 0.24999998224
117007 for threshold -0.375
```

In [161]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl1_3, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set3 Train data-L1")
```

Out[161]:

```
Text(0.5, 1.0, 'Confusion matrix -set3 Train data-L1')
```



In [162]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
17069 for threshold -0.085
[[ 3442  1985]
 [11797 18828]]
```

In [163]:

```
conf_matr_df_testl1_3 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

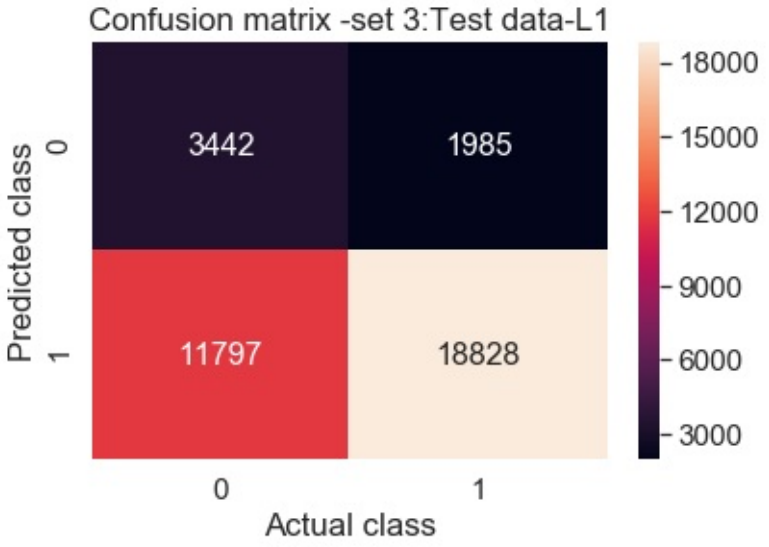
```
the maximum value of tpr*(1-fpr) 0.24999999151
17069 for threshold -0.085
```

In [164]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_testl1_3, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 3:Test data-L1")
```

Out[164]:

```
Text(0.5, 1.0, 'Confusion matrix -set 3:Test d
ata-L1')
```


Confusion matrix -set 3:Test data-L1
3442
1985
11797
18828
Predicted class
0
1
0
1
Actual class
18000
15000
12000
9000
6000
3000

Support Vector Machines on set 4

In [166]:

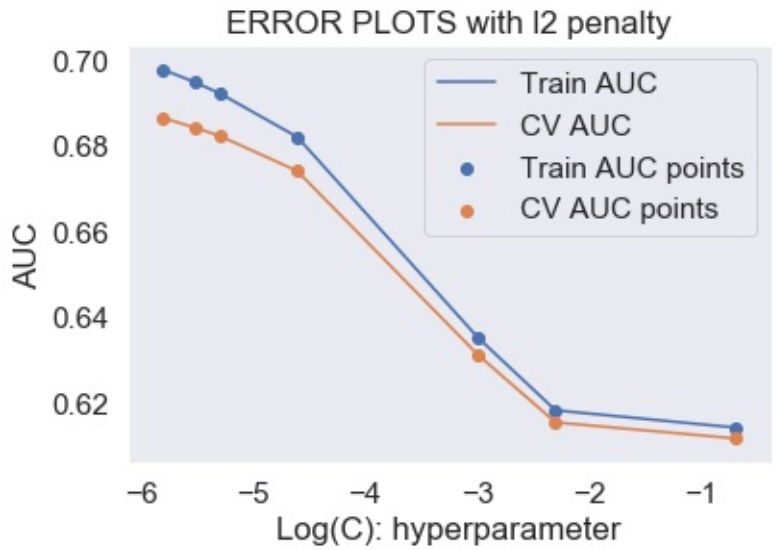
```
#Since we considered SGD classifier for linearsvm we need to
consider both hyperparameters L1 and L2 penalty
#and also hyperparemeter alpha of SGD is which is inverse of
C and no of points
##alpha=1/(C*m)
#C hypeparameter is regularization hyperparameter
# variables ready

X_tr=set4_train.tocsr()
X_cr=set4_cv.tocsr()
X_te=set4_test.tocsr()
```

Set4:doing Logistic regression with L2 penalty

In [167]:

```
%%time

#doing Logistic regression on L2 penalty
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005,0.004,0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha = i,penalty='
l2',random_state=42,class_weight ='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')
```

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l2 penalty")
plt.grid()
plt.show()
```




```
Wall time: 4 s
```

In [168]:

```
# We could see that the best hyperparameter for log(C) is -6
import math
k_best=math.pow(2.718281,-6)
```

In [169]:

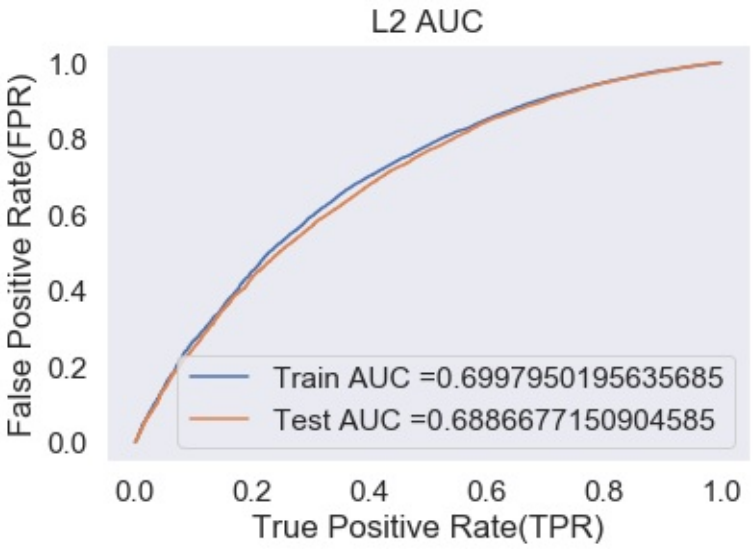
```
k_best
```

Out[169]:

0.0024787567094123678

In [170]:

```
# finding AUC for train and test for L2 penalty
from sklearn.metrics import roc_curve, auc


model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l2'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L2 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [171]:

```
def predict(proba, threshould, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [172]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold 0.131
[[ 3752  3752]
 [ 9076 32461]]
```

In [173]:

```
conf_matr_df_trainl2_4= pd.DataFrame(confusion_matrix(y_train
, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
, range(2),range(2))
```

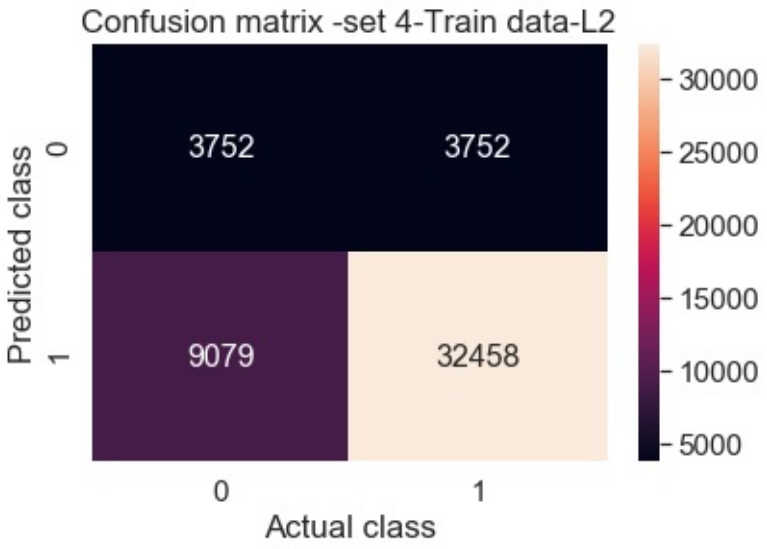
```
the maximum value of tpr*(1-fpr) 0.25 for thre
shold 0.131
```

In [145]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl2_4, annot=True,annot_kws={"si
ze": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 4-Train data-L2")
```

Out[145]:

```
Text(0.5, 1.0, 'Confusion matrix -set 4-Train
data-L2')
```



In [174]:

```
from sklearn.metrics import confusion_matrix
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold 0.393
[[ 3458  1969]
 [11104 19521]]
```

In [175]:

```
conf_matr_df_testl2_4 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_testl2_4, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
```

```
plt.title("Confusion matrix -set 4-Test data-L2")
```

```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold 0.393
```

Out[175]:

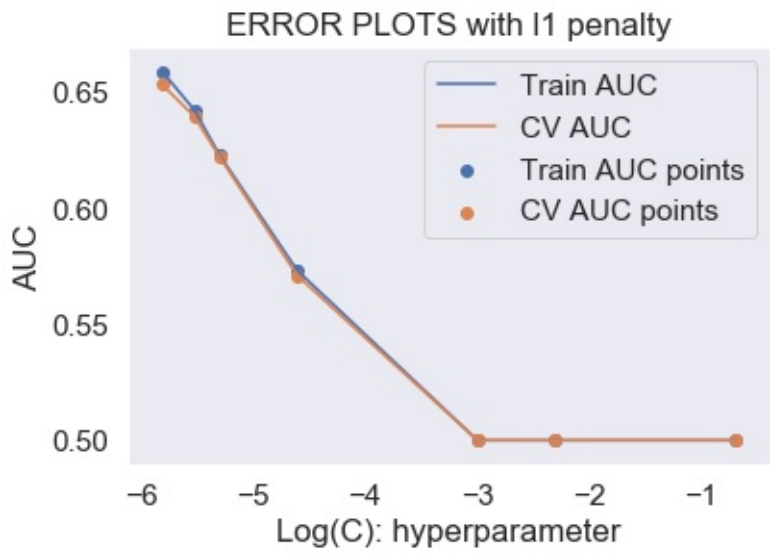
```
Text(0.5, 1.0, 'Confusion matrix -set 4-Test d
ata-L2')
```

Set4:doing SGD classification with L1 penalty

In [176]:

```
%%time
#doing Logistic regression on L1 penalty

import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import math
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes
]
Target scores, can either be probability estimates of the pos
itive class, confidence values, or non-thresholded measure of
decisions (as returned by “decision_function” on some classif
iers).
For binary y_true, y_score is supposed to be the score of the
 class with greater label.
"""
train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005, 0.004, 0.003]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha= i,penalty='l
1',random_state=42,class_weight='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
```

```
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l1 penalty")
plt.grid()
plt.show()
```




```
Wall time: 9.21 s
```

In [177]:

```
# We could see that the best hyperparameter for log(C) is -6
 for l1 penalty
import math
k_best=math.pow(2.718281,-6)
```

In [178]:

```
k_best
```

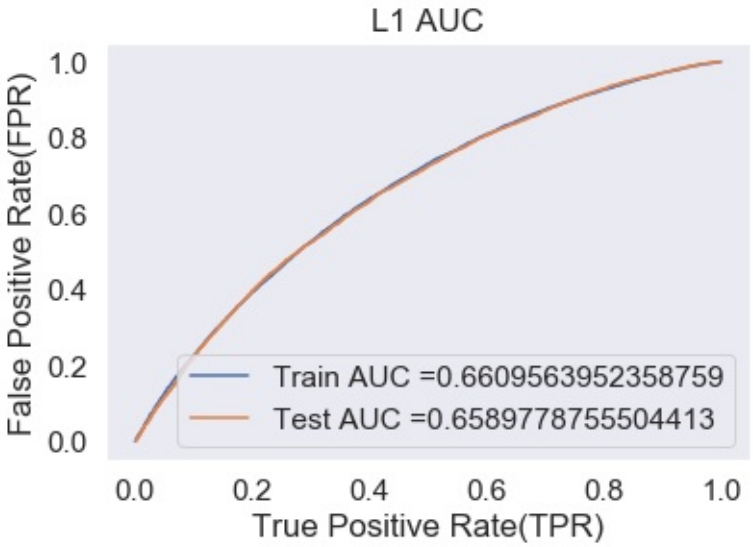
Out[178]:

```
0.0024787567094123678
```

In [179]:

```
# finding AUC for train and test for L1 penalty

from sklearn.metrics import roc_curve, auc

model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l1'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
```

```
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L1 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [180]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.279
[[ 3752  3752]
```

```
[11188 30349]]
```

In [181]:

```
conf_matr_df_trainl1_4 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

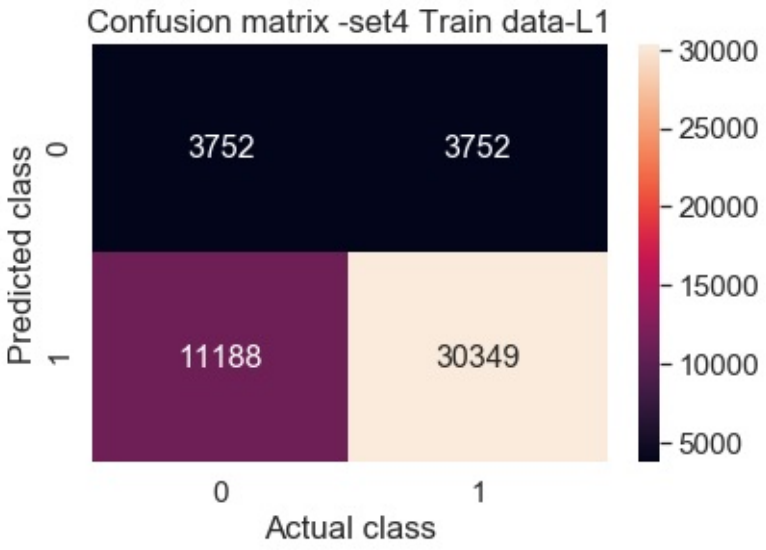
```
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.279
```

In [182]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl1_4, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set4 Train data-L1")
```

Out[182]:

```
Text(0.5, 1.0, 'Confusion matrix -set4 Train d
ata-L1')
```



In [183]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.065
[[ 3486  1941]
 [12577 18048]]
```

In [184]:

```
conf_matr_df_testl1_4 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

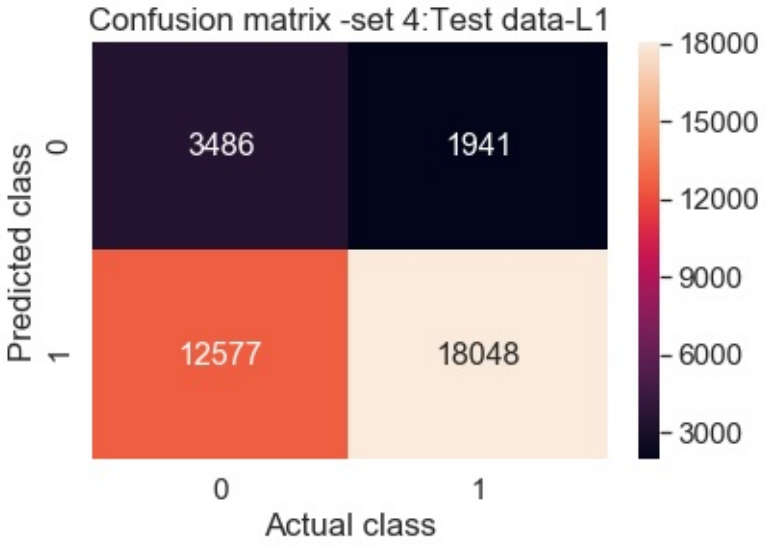
```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold -0.065
```

In [185]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_testl1_4, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 4:Test data-L1")
```

Out[185]:

```
Text(0.5, 1.0, 'Confusion matrix -set 4:Test d
ata-L1')
```


Confusion matrix -set 4:Test data-L1
3486
1941
12577
18048
Predicted class
0
1
Actual class
0
1
18000
15000
12000
9000
6000
3000

2.5 Support Vector Machines with added Features `Set 5`

In [186]:

```
#we need to perform truncated SVD on TFIDF on essay text data

#text_tfidf_train
#text_tfidf_test=
#text_tfidf_cv =
text_tfidf_train.shape
```

Out[186]:

```
(49041, 7500)
```

In [167]:

```
from sklearn.decomposition import TruncatedSVD

index = [5,10,50,100,250,500,1000,2500,5000]
variance_sum = []

for i in tqdm(index):
    svd = TruncatedSVD(n_components= i, n_iter=7, random_stat
e=42)
    svd.fit(text_tfidf_train)
    variance_sum.append(svd.explained_variance_ratio_.sum())
```

```
  0%|          | 0/9 [00:00<?, ?it/s]

 11%|▉         | 1/9 [00:05<00:45,  5.66s/it]

 22%|██        | 2/9 [00:07<00:31,  4.56s/it]
```

```
 33%|███       | 3/9 [00:12<00:27,  4.52s/it]

 44%|████      | 4/9 [00:20<00:27,  5.55s/it]

 56%|█████     | 5/9 [00:40<00:39,  9.95s/it]

 67%|██████    | 6/9 [01:22<00:58, 19.64s/it]

 78%|███████   | 7/9 [03:22<01:39, 49.88s/it]

 89%|████████  | 8/9 [12:29<03:18, 198.91s/it]

100%|██████████| 9/9 [1:02:10<00:00, 1032.39s/
it]
```

In [188]:

```
variance_sum
```

Out[188]:

```
[0.026996906595713455,
 0.04566409530168829,
 0.11774446445320151,
 0.17213272580432817,
 0.277820404998782,
 0.39141555019408547,
 0.532765901903532,
 0.7531189250934278,
 0.9245124584548546]
```

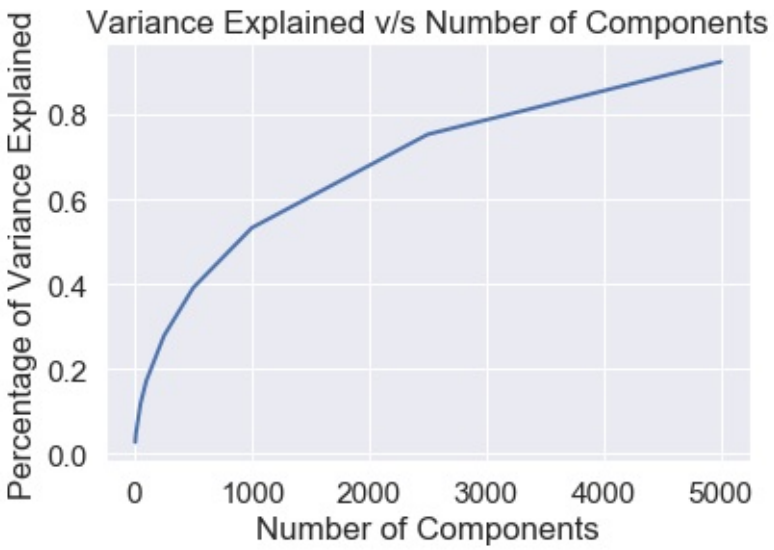
In [190]:

```
index = [5,10,50,100,250,500,1000,2500,5000]
```

In [191]:

```
import matplotlib.pyplot as plt
plt.xlabel("Number of Components")
```

```
plt.ylabel("Percentage of Variance Explained")
plt.title("Variance Explained v/s Number of Components")
plt.plot(index,variance_sum,lw=2)
plt.show()
```



In [192]:

```
print("Let us consider 5000 points as the number of Component
s. It Explains more than 90% of the Variance in the data")
```

```
Let us consider 5000 points as the number of C
omponents. It Explains more than 90% of the Va
riance in the data
```

In [194]:

```
# loading svd train test and cv values from pickle file
import pickle as pickle
#with open('C:/Users/pramod reddy chandi/Desktop/pram/applied
 ai course/DonorsChoose_2018/cat_num.pckl', 'rb') as f:
f=open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai
course/DonorsChoose_2018/svm_svd.pckl','rb')
svd_train,svd_test,svd_cv=pickle.load(f)
f.close()
```

In [101]:

```
from sklearn.decomposition import TruncatedSVD
svd.fit(text_tfidf_train)
svd_train = svd.transform(text_tfidf_train)

print("Shape of matrix after Decomposition ",svd_train.shape)
```

```
Shape of matrix after Decomposition  (49041, 5
000)
```

In [103]:

```
svd_test = svd.transform(text_tfidf_test)
print("Shape of matrix after Decomposition ",svd_test.shape)
```

```
Shape of matrix after Decomposition  (36052, 5
000)
```

In [104]:

```
svd_cv = svd.transform(text_tfidf_cv)
print("Shape of matrix after Decomposition ",svd_cv.shape)
```

```
Shape of matrix after Decomposition  (24155, 5
000)
```

In [110]:

```
import pickle
f=open('svm_svd.pckl','wb')
pickle.dump([svd_train,svd_test,svd_cv],f)
f.close()
```

In [195]:

```
svd_train.shape
```

Out[195]:

(49041, 5000)

In [198]:

```
svd_test.shape
```

Out[198]:

(36052, 5000)

In [197]:

```
svd_cv.shape
```

Out[197]:

(24155, 5000)

In [199]:

```
type(svd_train)
```

Out[199]:

numpy.ndarray

In [200]:

```
type(cat_num_train)
```

Out[200]:

scipy.sparse.coo.coo_matrix

In [201]:

```
# as it is difficult to convert nd array to sparse matrix we
wpuld convert sparse to dense and apply concatenation
cat_train=cat_num_train.todense()
cat_test=cat_num_test.todense()
cat_cv=cat_num_cv.todense()
```

In [202]:

```
import numpy as np
A=np.concatenate((svd_train,cat_train),axis=1)
```

In [203]:

```
B=np.concatenate((svd_test,cat_test),axis=1)
```

In [204]:

```
C=np.concatenate((svd_cv,cat_cv),axis=1)
```

In [205]:

```
A.shape
```

Out[205]:

```
(49041, 5108)
```

In [13]:

```
''''import pickle as pickle
#with open('C:/Users/pramod reddy chandi/Desktop/pram/applied
 ai course/DonorsChoose_2018/cat_num.pckl', 'rb') as f:
f=open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai
course/DonorsChoose_2018/y_values.pckl','rb')
y_train,y_test,y_cv=pickle.load(f)
f.close()'''
```

In [206]:

```
#making variables ready
X_tr=A
X_te=B
X_cr=C
```

In [207]:

```
X_te.shape
```

Out[207]:

```
(36052, 5108)
```

In [208]:

```
y_test.shape
```

Out[208]:

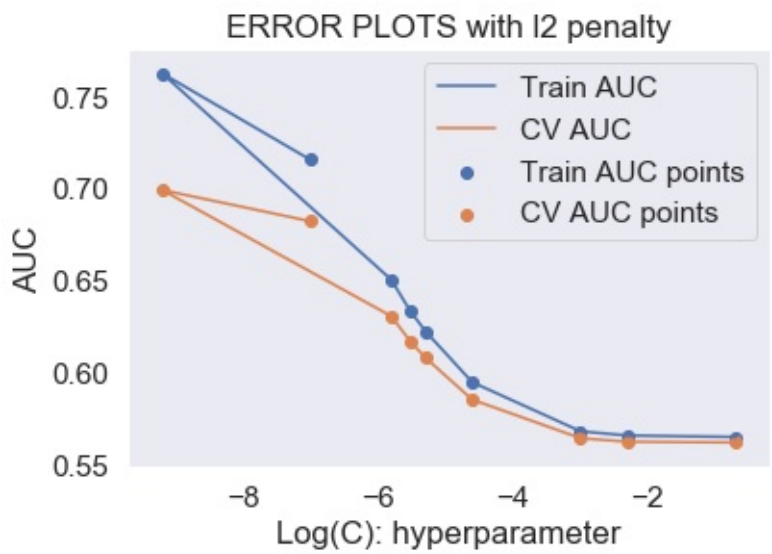
```
(36052,)
```

Set5:doing Logistic regression with L2 penalty

In [235]:

```
#Since we considered SGD classifier for linearsvm we need to
consider both hyperparameters L1 and L2 penalty
#and also hyperparemeter alpha of SGD is which is inverse of
C and no of points
##alpha=1/(C*m)
#C hypeparameter is regularization hyperparameter
# variables ready
#doing Logistic regression on L2 penalty
import matplotlib.pyplot as plt
from sklearn.linear_model import SGDClassifier
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_parameter=[]
K = [0.5, 0.1, 0.05, 0.01, 0.005,0.004,0.003,0.0001,0.0009]
for i in K:
    classifier=SGDClassifier(loss='hinge',alpha = i,penalty='
l2',random_state=42,class_weight='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
```

```
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l2 penalty")
plt.grid()
plt.show()
```



In [236]:

```
# We could see that the best hyperparameter for log(C) is -9
import math
k_best=math.pow(2.718281,-9)
```

In [237]:

```
k_best
```

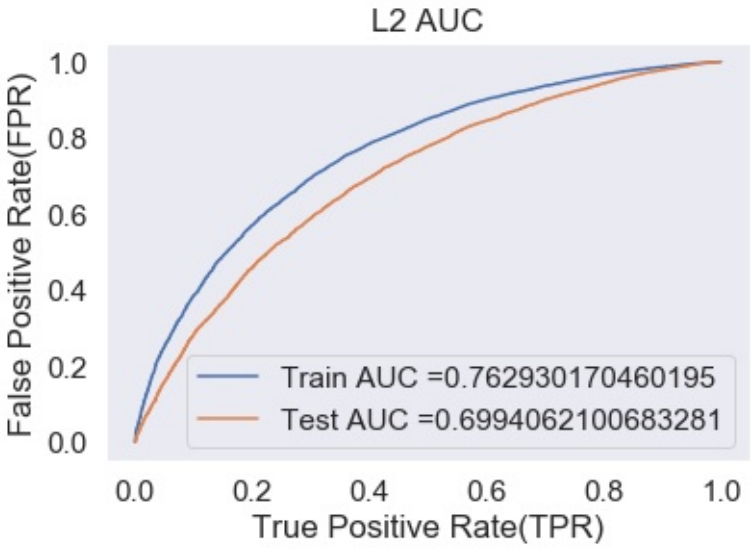
Out[237]:

```
0.00012341014259503752
```

In [238]:

```
# finding AUC for train and test for L2 penalty
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import SGDClassifier
import matplotlib.pyplot as plt

model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l2'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L2 AUC")
```

```
plt.grid()
plt.show()
```



Confusion matrix

In [239]:

```
def predict(proba, threshould, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
```

```
    return predictions
```

In [240]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.091
[[ 3752  3752]
 [ 6255 35282]]
```

In [241]:

```
import pandas as pd
conf_matr_df_trainl2_5= pd.DataFrame(confusion_matrix(y_train
, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
, range(2),range(2))
```

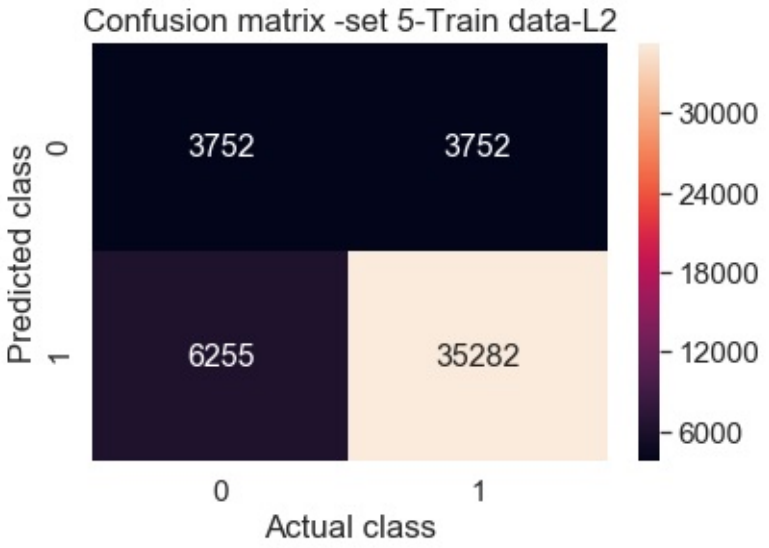
```
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.091
```

In [242]:

```
import seaborn as sns
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl2_5, annot=True,annot_kws={"si
ze": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 5-Train data-L2")
```

Out[242]:

```
Text(0.5, 1.0, 'Confusion matrix -set 5-Train
data-L2')
```



In [243]:

```
from sklearn.metrics import confusion_matrix
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold 0.315
[[ 3124  2303]
 [ 8580 22045]]
```

In [244]:

```
conf_matr_df_testl2_5 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold 0.315
```

In [245]:

```
sns.set(font_scale=1.4)#for label size
```

```
sns.heatmap(conf_matr_df_testl2_5, annot=True,annot_kws={"size": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 5-Test data-L2")
```

Out[245]:

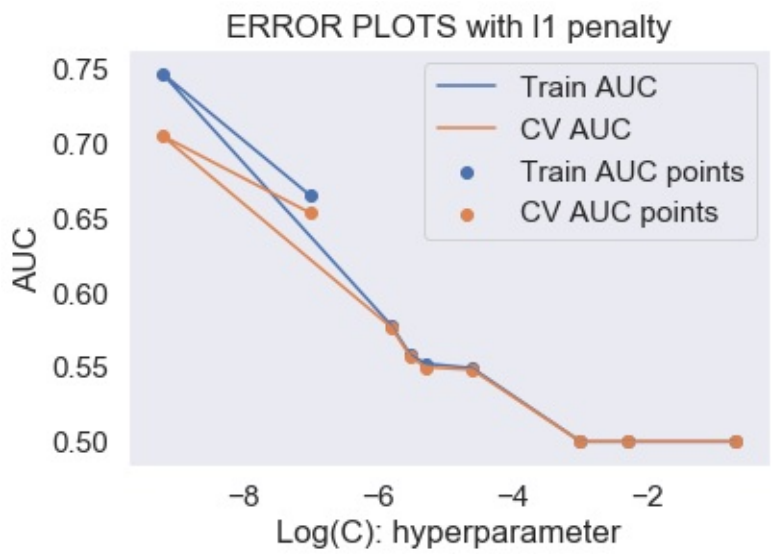
```
Text(0.5, 1.0, 'Confusion matrix -set 5-Test data-L2')
```

Set5:doing SGD classification with L1 penalty

In [246]:

```
#doing Logistic regression on L1 penalty

import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import math
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
y_score : array, shape = [n_samples] or [n_samples, n_classes
]
Target scores, can either be probability estimates of the pos
itive class, confidence values, or non-thresholded measure of
decisions (as returned by “decision_function” on some classif
iers).
For binary y_true, y_score is supposed to be the score of the
 class with greater label.
"""

train_auc = []
cv_auc = []
log_parameter=[]

K = [0.5, 0.1, 0.05, 0.01, 0.005, 0.004, 0.003,0.0001, 0.0009
]

for i in K:
    classifier=SGDClassifier(loss='hinge',alpha= i,penalty='l
```

```
1',random_state=42,class_weight='balanced',n_jobs=-1)
    classifier.fit(X_tr, y_train)
    y_train_pred = classifier.decision_function(X_tr)
    y_cv_pred = classifier.decision_function(X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    log_parameter.append(math.log(i))

plt.plot(log_parameter, train_auc, label='Train AUC')
plt.plot(log_parameter, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS with l1 penalty")
plt.grid()
plt.show()  bb ṁ
```



In [247]:

```
# We could see that the best hyperparameter for log(C) is -9
for l1 penalty
import math
k_best=math.pow(2.718281,-9)
```

In [248]:

```
k_best
```

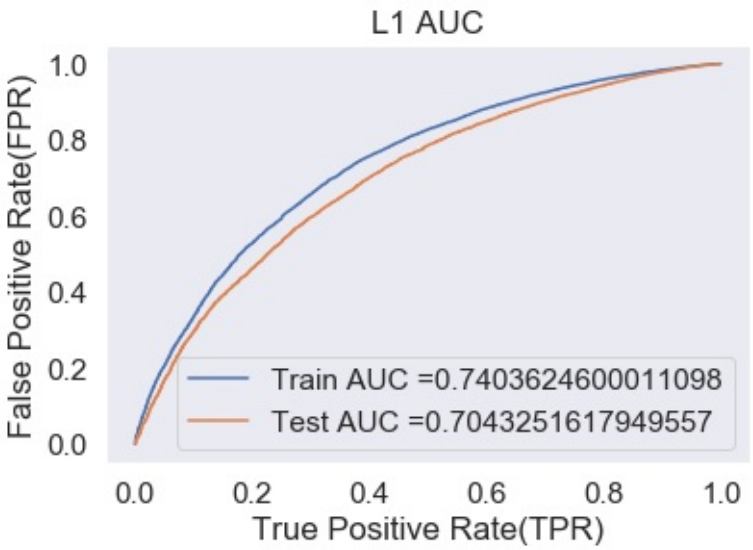
Out[248]:

```
0.00012341014259503752
```

In [249]:

```
# finding AUC for train and test for L1 penalty

from sklearn.metrics import roc_curve, auc

model = SGDClassifier(loss='hinge',alpha= k_best,penalty='l1'
,random_state=42,class_weight='balanced',n_jobs=-1)
model.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be
```

```
probability estimates of the positive class
# not the predicted outputs

y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("L1 AUC")
plt.grid()
plt.show()
```



Confusion matrix

In [250]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thre
sholds, train_fpr, train_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.007
[[ 3752  3752]
 [ 7191 34346]]
```

In [251]:

```
conf_matr_df_trainl1_5 = pd.DataFrame(confusion_matrix(y_trai
n, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)
), range(2),range(2))
```

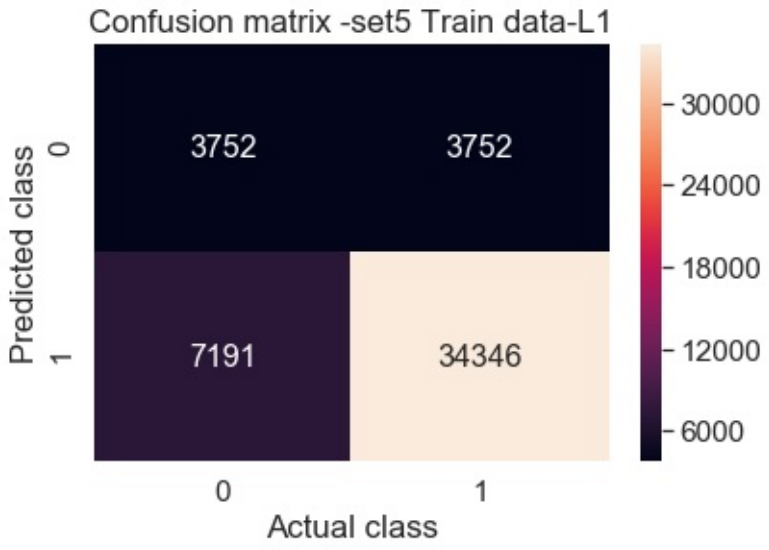
```
the maximum value of tpr*(1-fpr) 0.25 for thre
shold -0.007
```

In [252]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_trainl1_5, annot=True,annot_kws={"si
ze": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set5 Train data-L1")
```

Out[252]:

```
Text(0.5, 1.0, 'Confusion matrix -set5 Train d
ata-L1')
```



In [253]:

```
from sklearn.metrics import confusion_matrix

print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
olds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold 0.36
[[ 3280  2147]
 [ 9312 21313]]
```

In [254]:

```
conf_matr_df_testl1_5 = pd.DataFrame(confusion_matrix(y_test,
 predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), ra
nge(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999999151
170693 for threshold 0.36
```

In [255]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_testl1_5, annot=True,annot_kws={"siz
e": 16}, fmt='g')

plt.xlabel("Actual class")
plt.ylabel("Predicted class")
plt.title("Confusion matrix -set 5:Test data-L1")
```

Out[255]:

```
Text(0.5, 1.0, 'Confusion matrix -set 5:Test d
ata-L1')
```

3. Conclusion

In [257]:

```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable
 using: pip3 install prettytable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Penalty", "Alpha:Hyp
er Parameter", "Train AUC","Test AUC"]

x.add_row(["BOW", "Linear SVM","L1", 0.011, 0.63, 0.63])
x.add_row(["BOW", "Linear SVM","L2", 0.049, 0.76, 0.71])

x.add_row(["TFIDF", "Linear SVM", "L1", 0.002, 0.58, 0.57])
x.add_row(["TFIDF", "Linear SVM", "L2",0.002, 0.66 ,0.64])

x.add_row(["AVG W2V", "Linear SVM", "L1", 0.002, 0.68, 0.68])
x.add_row(["AVG W2V", "Linear SVM", "L2", 0.004, 0.68, 0.67])

x.add_row(["TFIDF W2V", "Linear SVM", "L1", 0.002, 0.66, 0.65
])
x.add_row(["TFIDF W2V", "Linear SVM", "L2", 0.002, 0.69, 0.68
])

x.add_row(["TRUNCATED SVD", "Linear SVM", "L1", 0.0009, 0.74,
 0.70 ])
x.add_row(["TRUNCATED SVD", "Linear SVM", "L2", 0.0009, 0.76,
 0.69 ])
```

```
print(x)
```

```
+---------------+------------+---------+------
-----------------+-----------+----------+
|   Vectorizer  |   Model    | Penalty | Alpha
:Hyper Parameter | Train AUC | Test AUC |
+---------------+------------+---------+------
-----------------+-----------+----------+
|      BOW      | Linear SVM |    L1   |
   0.011         |    0.63   |   0.63   |
|      BOW      | Linear SVM |    L2   |
   0.049         |    0.76   |   0.71   |
|     TFIDF     | Linear SVM |    L1   |
   0.002         |    0.58   |   0.57   |
|     TFIDF     | Linear SVM |    L2   |
   0.002         |    0.66   |   0.64   |
|    AVG W2V    | Linear SVM |    L1   |
   0.002         |    0.68   |   0.68   |
|    AVG W2V    | Linear SVM |    L2   |
   0.004         |    0.68   |   0.67   |
|   TFIDF W2V   | Linear SVM |    L1   |
   0.002         |    0.66   |   0.65   |
|   TFIDF W2V   | Linear SVM |    L2   |
   0.002         |    0.69   |   0.68   |
| TRUNCATED SVD | Linear SVM |    L1   |
   0.0009        |    0.74   |   0.7    |
| TRUNCATED SVD | Linear SVM |    L2   |
   0.0009        |    0.76   |   0.69   |
+---------------+------------+---------+------
-----------------+-----------+----------+
```

conclsuion : We can observe that truncated svd values gives best result it indicates that only most important features contribute lot to the predicted model.We can observe that even from results.