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**
`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 targete enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories f 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 ((https://en.wikipedia.org/wiki/List_of_U.S._state_abbr **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategori **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the projec • `My students need hands on literacy mater sensory needs!`

Feature	Description
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. **Ex** 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed pro bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated value • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted b **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 chart_studio.plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.1 Reading Data

In [2]:

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

In [3]:

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

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

In [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

1.2 preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

PREPROCESSING PROJECT GRADE CATEGORY

In [6]:

```
grade_categories=list(project_data['project_grade_category'].values)
clean_grades=[]
for i in grade_categories:
    temp=""
    for j in i.split(','):
        j=j.replace(' ','_')
        j=j.replace('-','_')
        temp+=j
        clean_grades.append(temp)
project_data['clean_grades']=clean_grades
project_data.drop(['project_grade_category'],axis=1,inplace=True)
```

In [7]:

```
clean_grades[0:5]
```

Out[7]:

```
['Grades_PreK_2', 'Grades_6_8', 'Grades_6_8', 'Grades_PreK_2', 'Grades_Pre
K_2']
```

In [8]:

```
project_data['clean_grades'].unique()
```

Out[8]:

```
array(['Grades_PreK_2', 'Grades_6_8', 'Grades_3_5', 'Grades_9_12'],
      dtype=object)
```

PREPROCESSING PROJECT SUBJECT CATEGORIES

In [9]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

1.3 Text preprocessing

In [10]:

```
# 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 [11]:

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

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

In [12]:

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

In [13]:

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

My students are English learners that are working on English as their seco
nd or third languages. We are a melting pot of refugees, immigrants, and n
ative-born Americans bringing the gift of language to our school. \r\n\r\n
We have over 24 languages represented in our English Learner program with
students at every level of mastery. We also have over 40 countries repres
ented with the families within our school. Each student brings a wealth o
f knowledge and experiences to us that open our eyes to new cultures, beli
efs, and respect.\"The limits of your language are the limits of your worl
d.\"-Ludwig Wittgenstein Our English learner's have a strong support syst
em at home that begs for more resources. Many times our parents are learn
ing to read and speak English along side of their children. Sometimes thi
s creates barriers for parents to be able to help their child learn phonet
ics, letter recognition, and other reading skills.\r\n\r\nBy providing the
se dvd's and players, students are able to continue their mastery of the E
nglish language even if no one at home is able to assist. All families wi
th students within the Level 1 proficiency status, will be a offered to be
a part of this program. These educational videos will be specially chosen
by the English Learner Teacher and will be sent home regularly to watch.
The videos are to help the child develop early reading skills.\r\n\r\nPare
nts that do not have access to a dvd player will have the opportunity to c
heck out a dvd player to use for the year. The plan is to use these video
s and educational dvd's for the years to come for other EL students.\r\nna
nnan

==

The 51 fifth grade students that will cycle through my classroom this year
all love learning, at least most of the time. At our school, 97.3% of the
students receive free or reduced price lunch. Of the 560 students, 97.3% a
re minority students. \r\nThe school has a vibrant community that loves to
get together and celebrate. Around Halloween there is a whole school parad
e to show off the beautiful costumes that students wear. On Cinco de Mayo
we put on a big festival with crafts made by the students, dances, and gam
es. At the end of the year the school hosts a carnival to celebrate the ha
rd work put in during the school year, with a dunk tank being the most pop
ular activity.My students will use these five brightly colored Hokki stool
s in place of regular, stationary, 4-legged chairs. As I will only have a
total of ten in the classroom and not enough for each student to have an i
ndividual one, they will be used in a variety of ways. During independent
reading time they will be used as special chairs students will each use on
occasion. I will utilize them in place of chairs at my small group tables
during math and reading times. The rest of the day they will be used by th
e students who need the highest amount of movement in their life in order
to stay focused on school.\r\n\r\nWhenever asked what the classroom is mis
sing, my students always say more Hokki Stools. They can't get their fill
of the 5 stools we already have. When the students are sitting in group wi
th me on the Hokki Stools, they are always moving, but at the same time do
ing their work. Anytime the students get to pick where they can sit, the H
okki Stools are the first to be taken. There are always students who head
over to the kidney table to get one of the stools who are disappointed as
there are not enough of them. \r\n\r\nWe ask a lot of students to sit for
7 hours a day. The Hokki stools will be a compromise that allow my student
s to do desk work and move at the same time. These stools will help studen
ts to meet their 60 minutes a day of movement by allowing them to activate
their core muscles for balance while they sit. For many of my students, th
ese chairs will take away the barrier that exists in schools for a child w
ho can't sit still.nannan

==

How do you remember your days of school? Was it in a sterile environment w
ith plain walls, rows of desks, and a teacher in front of the room? A typi
cal day in our room is nothing like that. I work hard to create a warm inv
iting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed r

aces in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

==

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

==

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students are on free or reduced lunch. We aren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character.In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the blue tooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use. The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.nannan

==

In [14]:

```
# 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 [15]:

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

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. The
y are eager beavers and always strive to work their hardest working past t
heir limitations. \r\n\r\nThe materials we have are the ones I seek out fo
r my students. I teach in a Title I school where most of the students rece
ive free or reduced price lunch.  Despite their disabilities and limitatio
ns, my students love coming to school and come eager to learn and explore.
Have you ever felt like you had ants in your pants and you needed to groov
e and move as you were in a meeting? This is how my kids feel all the tim
e. The want to be able to move as they learn or so they say.Wobble chairs
are the answer and I love then because they develop their core, which enha
nces gross motor and in Turn fine motor skills. \r\nThey also want to lear
n through games, my kids do not want to sit and do worksheets. They want t
o learn to count by jumping and playing. Physical engagement is the key to
our success. The number toss and color and shape mats can make that happe
n. My students will forget they are doing work and just have the fun a 6 y
ear old deserves.nannan
==================================================
```

In [16]:

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

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

In [17]:

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

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

In [18]:

```
# 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', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [19]:

```
# 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)
    sent = sent.lower()
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████
████| 109248/109248 [01:55<00:00, 947.64it/s]
```

In [20]:

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

Out[20]:

```
'kindergarten students varied disabilities ranging speech language delays
cognitive delays gross fine motor delays autism eager beavers always striv
e work hardest working past limitations materials ones seek students teach
title school students receive free reduced price lunch despite disabilitie
s limitations students love coming school come eager learn explore ever fe
lt like ants pants needed groove move meeting kids feel time want able mov
e learn say wobble chairs answer love develop core enhances gross motor tu
rn fine motor skills also want learn games kids not want sit worksheets wa
nt learn count jumping playing physical engagement key success number toss
color shape mats make happen students forget work fun 6 year old deserves
nannan'
```

1.4 Preprocessing of `project_title`

In [21]:

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

```
100%|██████████████████████████████████████████████████████████████████████
██| 109248/109248 [00:05<00:00, 20648.15it/s]
```

1.5 Preparing data for models

In [22]:

```
project_data.columns
```

Out[22]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_grades', 'clean_subcategories', 'essa
y'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [23]:

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

```
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'lit
eracy_language', 'math_science', 'music_arts', 'specialneeds', 'warmth']
Shape of matrix after one hot encodig  (109248, 9)
```

In [24]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer()
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].v
alues)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_governmen
t', 'college_careerprep', 'communityservice', 'earlydevelopment', 'economi
cs', 'environmentalscience', 'esl', 'extracurricular', 'financialliterac
y', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_welln
ess', 'history_geography', 'literacy', 'literature_writing', 'mathematic
s', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'performi
ngarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'wa
rmth']
Shape of matrix after one hot encodig  (109248, 30)
```

In [25]:

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

In [26]:

```
#vectorizing school state
vectorizer=CountVectorizer(vocabulary=project_data['school_state'].unique(),lowercase=F
alse,binary=True)
school_state_one_hot=vectorizer.fit_transform(project_data['school_state'].values)
print(vectorizer.get_feature_names())
print('shape of matrix after one ot encoding',school_state_one_hot.shape)
```

```
['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY', 'OK',
'MA', 'NV', 'OH', 'PA', 'AL', 'LA', 'VA', 'AR', 'WA', 'WV', 'ID', 'TN', 'M
S', 'CO', 'UT', 'IL', 'MI', 'HI', 'IA', 'RI', 'NJ', 'MO', 'DE', 'MN', 'M
E', 'WY', 'ND', 'OR', 'AK', 'MD', 'WI', 'SD', 'NE', 'NM', 'DC', 'KS', 'M
T', 'NH', 'VT']
shape of matrix after one ot encoding (109248, 51)
```

In [27]:

```
#vectorizing teacher prefix
x=project_data['teacher_prefix'].fillna('')
vectorizer = CountVectorizer(vocabulary=x.unique(), lowercase=False, binary=True)

teacher_prefix_one_hot = vectorizer.fit_transform(x.values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot.shape)
```

```
['Mrs.', 'Mr.', 'Ms.', 'Teacher', '', 'Dr.']
Shape of matrix after one hot encodig  (109248, 6)
```

In [28]:

```
#vectorizing project grade category
vectorizer = CountVectorizer(vocabulary=project_data['clean_grades'].unique(), lowercas
e=False, binary=True)
project_grade_one_hot = vectorizer.fit_transform(project_data['clean_grades'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",project_grade_one_hot.shape)
```

```
['Grades_PreK_2', 'Grades_6_8', 'Grades_3_5', 'Grades_9_12']
Shape of matrix after one hot encodig  (109248, 4)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [29]:

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

```
Shape of matrix after one hot encodig  (109248, 16512)
```

In [30]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer=CountVectorizer(min_df=10)
title_bow=vectorizer.fit_transform(preprocessed_titles)
print("shape of matrix after encoding",title_bow.shape)
```

```
shape of matrix after encoding (109248, 3222)
```

1.5.2.2 TFIDF vectorizer

In [31]:

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

```
Shape of matrix after one hot encoding  (109248, 16512)
```

In [32]:

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

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

1.5.2.3 Using Pretrained Models: Avg W2V

In [33]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-to-save-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)


'''
```

Out[33]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Mode
l")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    f
or line in tqdm(f):\n        splitLine = line.split()\n        word = spli
tLine[0]\n        embedding = np.array([float(val) for val in splitLine
[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," w
ords loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.
txt\')\n\n# ============================\nOutput:\n    \nLoading Glove Mod
el\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ====
========================\n\nwords = []\nfor i in preproced_texts:\n    wor
ds.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend
(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords
= set(words)\nprint("the unique words in the coupus", len(words))\n\ninter
_words = set(model.keys()).intersection(words)\nprint("The number of words
that are present in both glove vectors and our coupus",       len(inter_wo
rds),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpu
s = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in word
s_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length",
len(words_courpus))\n\n\n\n# stronging variables into pickle files python: h
ttp://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-
python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n
pickle.dump(words_courpus, f)\n\n\n'
```

In [34]:

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

In [0]:

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

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

```
100%|██████████████████████████████████████████████████████████████████
███| 109248/109248 [01:00<00:00, 1806.88it/s]

109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [0]:

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

In [0]:

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

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

```
100%|████████████████████████████████████████████████████████████████
████| 109248/109248 [08:03<00:00, 225.81it/s]

109248
300
```

In [0]:

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

1.5.3 Vectorizing Numerical features

In [33]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_i
ndex()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [48]:

```
data1=project_data.drop(['id','teacher_id','project_essay_1','project_essay_2','project
_essay_3','project_essay_4','project_is_approved'],axis=1)
data1.head(2)
data=data1[0:100000]
data[0:2]
```

Out[48]:

	Unnamed: 0	teacher_prefix	school_state	project_submitted_datetime	project_title
0	160221	Mrs.	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home
1	140945	Mr.	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners

In [35]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ...
 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

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

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

In [36]:

```
price_standardized
```

Out[36]:

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

In [37]:

```
projects_scalar = StandardScaler()
projects_scalar.fit(project_data['teacher_number_of_previously_posted_projects'].values
.reshape(-1,1)) # finding the mean and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_
[0])}")

# Now standardize the data with above maen and variance.
projects_standardized = projects_scalar.transform(project_data['teacher_number_of_previ
ously_posted_projects'].values.reshape(-1,1))
projects_standardized
```

```
C:\Users\HP\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ut
ils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.


Mean : 298.1193425966608, Standard deviation : 367.49634838483496

C:\Users\HP\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\ut
ils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

Out[37]:

```
array([[-0.40152481],
       [-0.14951799],
       [-0.36552384],
       ...,
       [-0.29352189],
       [-0.40152481],
       [-0.40152481]])
```

1.5.4 Merging all the above features

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

In [38]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(teacher_prefix_one_hot.shape)
print(school_state_one_hot.shape)
print(project_grade_one_hot.shape)
print(title_bow.shape)
print(text_bow.shape)
print(price_standardized.shape)
print(projects_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 6)
(109248, 51)
(109248, 4)
(109248, 3222)
(109248, 16512)
(109248, 1)
(109248, 1)
```

In [39]:

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

Out[39]:

```
(109248, 19836)
```

In [47]:

```
y1=project_data['project_is_approved']
print(y1.shape)
y=y1[0:100000]
```

```
(109248,)
```

In [41]:

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

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes 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)

2. **The hyper paramter tuning(find best Alpha)**

   - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
   - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
   - Find the best hyper paramter using k-fold cross validation or simple cross validation data
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Feature importance**

   - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names

4. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.

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

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

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

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

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

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

   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

2. Naive Bayes

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

In [49]:

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

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(data,y,test_size=0.30,stratify=y)
X_train,X_cv,y_train,y_cv=train_test_split(X_train,y_train,test_size=0.30,stratify=y_tr
ain)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [43]:

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

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

In [50]:

```
#encoding categorical variable school state
vectorizer=CountVectorizer()
vectorizer.fit(X_train['school_state'].values)

X_train_state_encoded=vectorizer.transform(X_train['school_state'].values)
X_cv_state_encoded=vectorizer.transform(X_cv['school_state'].values)
X_test_state_encoded=vectorizer.transform(X_test['school_state'].values)

print("AFTER VECTORIZATION")
print('='*50)
print(X_train_state_encoded.shape,y_train.shape)
print(X_cv_state_encoded.shape,y_cv.shape)
print(X_test_state_encoded.shape,y_test.shape)
```

```
AFTER VECTORIZATION
==================================================
(49000, 51) (49000,)
(21000, 51) (21000,)
(30000, 51) (30000,)
```

In [51]:

```
(project_data['teacher_prefix'].unique())
```

Out[51]:

```
array(['Mrs.', 'Mr.', 'Ms.', 'Teacher', nan, 'Dr.'], dtype=object)
```

In [52]:

```
X_train['teacher_prefix'].unique()
```

Out[52]:

```
array(['Mr.', 'Mrs.', 'Ms.', 'Teacher', 'Dr.', nan], dtype=object)
```

In [55]:

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

In [56]:

```
vectorizer=CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values)
X_train_prefix_encoded=vectorizer.transform(X_train['teacher_prefix'].values)
X_cv_prefix_encoded=vectorizer.transform(X_cv['teacher_prefix'].values)
X_test_prefix_encoded=vectorizer.transform(X_test['teacher_prefix'].values)

print('AFTER VECTORIZATION')
print('='*50)
print(vectorizer.get_feature_names())
print(X_train_prefix_encoded.shape,y_train.shape)
print(X_cv_prefix_encoded.shape,y_cv.shape)
print(X_test_prefix_encoded.shape,y_test.shape)
```

```
AFTER VECTORIZATION
==================================================
['dr', 'mr', 'mrs', 'ms', 'teacher']
(49000, 5) (49000,)
(21000, 5) (21000,)
(30000, 5) (30000,)
```

In [57]:

```
#encoding project grade category
vectorizer=CountVectorizer()
vectorizer.fit(X_train['clean_grades'].values)
X_train_grade_encoded=vectorizer.transform(X_train['clean_grades'].values)
X_cv_grade_encoded=vectorizer.transform(X_cv['clean_grades'].values)
X_test_grade_encoded=vectorizer.transform(X_test['clean_grades'].values)

print('AFTER VECTORIZATION')
print('='*50)
print(X_train_grade_encoded.shape,y_train.shape)
print(X_cv_grade_encoded.shape,y_cv.shape)
print(X_test_grade_encoded.shape,y_test.shape)
```

```
AFTER VECTORIZATION
==================================================
(49000, 4) (49000,)
(21000, 4) (21000,)
(30000, 4) (30000,)
```

In [58]:

```
#encoding clean categories
vectorizer=CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values)
X_train_categories_encoded=vectorizer.transform(X_train['clean_categories'].values)
X_cv_categories_encoded=vectorizer.transform(X_cv['clean_categories'].values)
X_test_categories_encoded=vectorizer.transform(X_test['clean_categories'].values)
print("AFTER VECTORIZATION")
print('='*50)
print(X_train_categories_encoded.shape,y_train.shape)
print(X_cv_categories_encoded.shape,y_cv.shape)
print(X_test_categories_encoded.shape,y_test.shape)
```

```
AFTER VECTORIZATION
==================================================
(49000, 9) (49000,)
(21000, 9) (21000,)
(30000, 9) (30000,)
```

In [59]:

```
#encoding subcategories
vectorizer=CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values)
X_train_subcategories_encoded=vectorizer.transform(X_train['clean_subcategories'].value
s)
X_cv_subcategories_encoded=vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_subcategories_encoded=vectorizer.transform(X_test['clean_subcategories'].values)
print("AFTER VECTORIZATION")
print('='*50)
print(X_train_subcategories_encoded.shape,y_train.shape)
print(X_cv_subcategories_encoded.shape,y_cv.shape)
print(X_test_subcategories_encoded.shape,y_test.shape)
```

```
AFTER VECTORIZATION
==================================================
(49000, 30) (49000,)
(21000, 30) (21000,)
(30000, 30) (30000,)
```

In [60]:

```
#encoding numerical categories---price
from sklearn.preprocessing import Normalizer
normalizer=Normalizer()
normalizer.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm=normalizer.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_norm=normalizer.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm=normalizer.transform(X_test['price'].values.reshape(-1,1))

print("after vectorization")
print(X_train_price_norm.shape,y_train.shape)
print(X_cv_price_norm.shape,y_cv.shape)
print(X_test_price_norm.shape,y_test.shape)
```

```
after vectorization
(49000, 1) (49000,)
(21000, 1) (21000,)
(30000, 1) (30000,)
```

In [61]:

```
#encoding numerical category quantity
normalizer=Normalizer()
normalizer.fit(X_train['quantity'].values.reshape(-1,1))

X_train_quantity_norm=normalizer.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_quantity_norm=normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quantity_norm=normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print('after vectorization')
print(X_train_quantity_norm.shape,y_train.shape)
print(X_cv_quantity_norm.shape,y_cv.shape)
print(X_test_quantity_norm.shape,y_test.shape)
```

```
after vectorization
(49000, 1) (49000,)
(21000, 1) (21000,)
(30000, 1) (30000,)
```

In [62]:

```
#encoding previous projects posted by teachers
normalizer=Normalizer()
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-
1,1))

X_train_projects_norm=normalizer.transform(X_train['teacher_number_of_previously_posted
_projects'].values.reshape(-1,1))
X_cv_projects_norm=normalizer.transform(X_cv['teacher_number_of_previously_posted_proje
cts'].values.reshape(-1,1))
X_test_projects_norm=normalizer.transform(X_test['teacher_number_of_previously_posted_p
rojects'].values.reshape(-1,1))

print("after vectorization")
print(X_train_projects_norm.shape,y_train.shape)
print(X_cv_projects_norm.shape,y_cv.shape)
print(X_test_projects_norm.shape,y_test.shape)
```

```
after vectorization
(49000, 1) (49000,)
(21000, 1) (21000,)
(30000, 1) (30000,)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [63]:

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

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

In [64]:

```
print(X_train.shape)
print(y_train.shape)
print(X_cv.shape)
print(y_cv.shape)
print(X_test.shape)
print(y_test.shape)
print('='*50)


vectorizer=CountVectorizer(min_df=10,ngram_range=(1,1))
vectorizer.fit(X_train['essay'].values)
X_train_essay_bow=vectorizer.transform(X_train['essay'].values)
#print(X_train_essay_bow.shape)
X_cv_essay_bow=vectorizer.transform(X_cv["essay"].values)
X_test_essay_bow=vectorizer.transform(X_test['essay'].values)

print('AFTER VECTORIZATION')
print('='*50)
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
```

```
(49000, 13)
(49000,)
(21000, 13)
(21000,)
(30000, 13)
(30000,)
==================================================
AFTER VECTORIZATION
==================================================
(49000, 12527) (49000,)
(21000, 12527) (21000,)
(30000, 12527) (30000,)
```

In [65]:

```
#encoding project title
vectorizer=CountVectorizer(min_df=10,ngram_range=(1,1))
vectorizer.fit(X_train['project_title'].values)
X_train_title_bow=vectorizer.transform(X_train['project_title'].values)
X_cv_title_bow=vectorizer.transform(X_cv['project_title'].values)
X_test_title_bow=vectorizer.transform(X_test['project_title'].values)
print("after vectorization")
print(X_train_title_bow.shape,y_train.shape)
print(X_cv_title_bow.shape,y_cv.shape)
print(X_test_title_bow.shape,y_test.shape)
```

```
after vectorization
(49000, 2108) (49000,)
(21000, 2108) (21000,)
(30000, 2108) (30000,)
```

2.4 Appling NB() on different kind of featurization as mentioned in the instructions

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

2.4.1 Applying Naive Bayes on BOW, SET 1

In [66]:

```
# Please write all the code with proper documentation
from scipy.sparse import hstack
final_train_bow=hstack((X_train_state_encoded,X_train_prefix_encoded,X_train_grade_enco
ded,X_train_categories_encoded,X_train_subcategories_encoded,X_train_price_norm,X_train
_projects_norm,X_train_quantity_norm,X_train_essay_bow,X_train_title_bow)).tocsr()
final_cv_bow=hstack((X_cv_state_encoded,X_cv_prefix_encoded,X_cv_grade_encoded,X_cv_cat
egories_encoded,X_cv_subcategories_encoded,X_cv_price_norm,X_cv_projects_norm,X_cv_quan
tity_norm,X_cv_essay_bow,X_cv_title_bow)).tocsr()
final_test_bow=hstack((X_test_state_encoded,X_test_prefix_encoded,X_test_grade_encoded,
X_test_categories_encoded,X_test_subcategories_encoded,X_test_price_norm,X_test_project
s_norm,X_test_quantity_norm,X_test_essay_bow,X_test_title_bow)).tocsr()
print(final_train_bow.shape,y_train.shape)
print(final_cv_bow.shape,y_cv.shape)
print(final_test_bow.shape,y_test.shape)
```

```
(49000, 14737) (49000,)
(21000, 14737) (21000,)
(30000, 14737) (30000,)
```

In [67]:

```
#creating final data matrix
from scipy.sparse import hstack
final_train_bow=hstack((X_train_state_encoded,X_train_prefix_encoded,X_train_grade_enco
ded,X_train_categories_encoded,X_train_subcategories_encoded,X_train_price_norm,X_train
_projects_norm,X_train_quantity_norm,X_train_essay_bow,X_train_title_bow)).tocsr()
final_cv_bow=hstack((X_cv_state_encoded,X_cv_prefix_encoded,X_cv_grade_encoded,X_cv_cat
egories_encoded,X_cv_subcategories_encoded,X_cv_price_norm,X_cv_projects_norm,X_cv_quan
tity_norm,X_cv_essay_bow,X_cv_title_bow)).tocsr()
final_test_bow=hstack((X_test_state_encoded,X_test_prefix_encoded,X_test_grade_encoded,
X_test_categories_encoded,X_test_subcategories_encoded,X_test_price_norm,X_test_project
s_norm,X_test_quantity_norm,X_test_essay_bow,X_test_title_bow)).tocsr()
print(final_train_bow.shape,y_train.shape)
print(final_cv_bow.shape,y_cv.shape)
print(final_test_bow.shape,y_test.shape)
```

```
(49000, 14737) (49000,)
(21000, 14737) (21000,)
(30000, 14737) (30000,)
```

In [73]:

```
#plotting error plots
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
from sklearn.svm.libsvm import predict_proba

train_auc=[]
cv_auc=[]
c=[0.00001,0.0001,0.001,0.01,0.1,1,5]
for i in tqdm(c):
    naive_bayes=MultinomialNB(alpha=i,class_prior=[0.5,0.5])
    naive_bayes.fit(final_train_bow,y_train)

    y_tr_pred=naive_bayes.predict_proba(final_train_bow)
    y_cv_pred=naive_bayes.predict_proba(final_cv_bow)

    x=roc_auc_score(y_train,y_tr_pred)
    y=roc_auc_score(y_cv,y_cv_pred)

    train_auc.extend(x)
    cv_auc.extend(y)
plt.plot(c,train_auc,label='TRAIN AUC')
plt.plot(c,cv_auc,label="CV AUC")
plt.scatter(c,train_auc,label='TRAIN AUC POINTS')
plt.scatter(c,cv_auc,label="CV AUC POINTS")
plt.legend()
plt.title("ERROR PLOTS")
plt.xlabel("alpha- HYPERPARAMETER")
plt.ylabel("auc score")
plt.grid()
plt.show()
```

```
  0%|
| 0/7 [00:00<?, ?it/s]
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-73-bc84789838d3> in <module>()
     15     y_cv_pred=naive_bayes.predict_proba(final_cv_bow)
     16
---> 17     x=roc_auc_score(y_train,y_tr_pred)
     18     y=roc_auc_score(y_cv,y_cv_pred)
     19

~\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\rank
ing.py in roc_auc_score(y_true, y_score, average, sample_weight)
    275     return _average_binary_score(
    276         _binary_roc_auc_score, y_true, y_score, average,
--> 277         sample_weight=sample_weight)
    278
    279

~\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\bas
e.py in _average_binary_score(binary_metric, y_true, y_score, average, sam
ple_weight)
     73
     74     if y_type == "binary":
---> 75         return binary_metric(y_true, y_score, sample_weight=sample
_weight)
     76
     77     check_consistent_length(y_true, y_score, sample_weight)

~\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\rank
ing.py in _binary_roc_auc_score(y_true, y_score, sample_weight)
    270
    271         fpr, tpr, tresholds = roc_curve(y_true, y_score,
--> 272                                         sample_weight=sample_weigh
t)
    273         return auc(fpr, tpr, reorder=True)
    274

~\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\rank
ing.py in roc_curve(y_true, y_score, pos_label, sample_weight, drop_interm
ediate)
    532     """
    533     fps, tps, thresholds = _binary_clf_curve(
--> 534         y_true, y_score, pos_label=pos_label, sample_weight=sample
_weight)
    535
    536     # Attempt to drop thresholds corresponding to points in betwee
n and

~\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\metrics\rank
ing.py in _binary_clf_curve(y_true, y_score, pos_label, sample_weight)
    320     check_consistent_length(y_true, y_score, sample_weight)
    321     y_true = column_or_1d(y_true)
--> 322     y_score = column_or_1d(y_score)
    323     assert_all_finite(y_true)
    324     assert_all_finite(y_score)

~\AppData\Local\Continuum\anaconda3\lib\site-packages\sklearn\utils\valida
tion.py in column_or_1d(y, warn)
    612         return np.ravel(y)
    613
```

```
--> 614     raise ValueError("bad input shape {0}".format(shape))
    615
    616

ValueError: bad input shape (49000, 2)
```

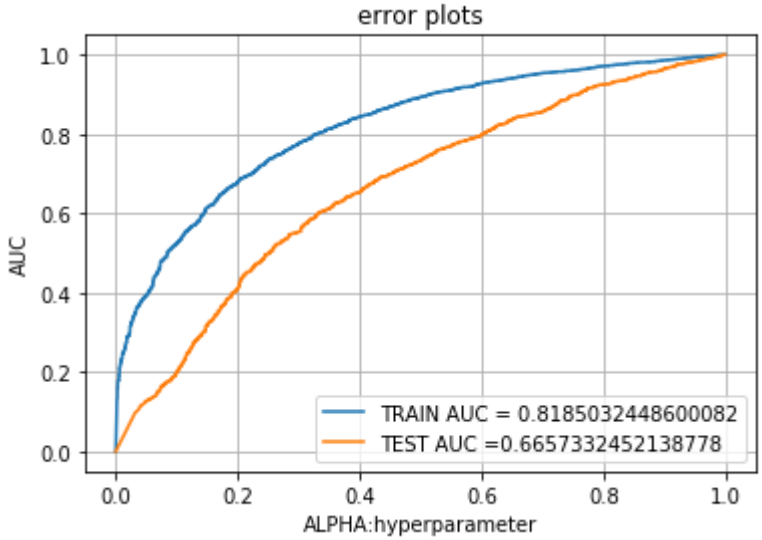
OBSERVATIONS:

Here we plotted for different values of alpha.
From those values we choose our best alpha to be 1.

In [89]:

```
from sklearn.metrics import roc_curve,auc

naive_bayes=MultinomialNB(alpha=np.log(1))
naive_bayes.fit(final_train_bow,y_train)

y_train_pred=batch_predict(naive_bayes,final_train_bow)
y_test_pred=batch_predict(naive_bayes,final_test_bow)

train_fpr,train_tpr,tr_threshold=roc_curve(y_train,y_train_pred)
test_fpr,test_tpr,test_threshold=roc_curve(y_test,y_test_pred)

plt.plot(train_fpr,train_tpr,label="TRAIN AUC = "+str(auc(train_fpr,train_tpr)))
plt.plot(test_fpr,test_tpr,label="TEST AUC ="+str(auc(test_fpr,test_tpr)))

plt.legend()
plt.xlabel('ALPHA:hyperparameter')
plt.ylabel('AUC')
plt.title("error plots")
plt.grid()
plt.show()
```

```
C:\Users\DELL\Anaconda3\lib\site-packages\sklearn\naive_bayes.py:480: User
Warning:

alpha too small will result in numeric errors, setting alpha = 1.0e-10
```



OBSERVATIONS:

Here we plotted naive bayes with alpha=log(1) as we choose our alpha to be 1.
We got train auc value of 81.85% and test auc value as 66.57%.

In [90]:

```
def find_best_threshold(threshold, fpr, tpr):
    t = threshold[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
nd(t,3))
    return t

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

In [91]:

```
#printing confusion matrix
print('='*100)
from sklearn.metrics import confusion_matrix
best_t=find_best_threshold(tr_threshold,train_fpr,train_tpr)
print('TRAIN CONFUSION MATRIX')
print(confusion_matrix(y_train,predict_with_best_t(y_train_pred,best_t)))
print('test confusion matrix')
print(confusion_matrix(y_test,predict_with_best_t(y_test_pred,best_t)))
```

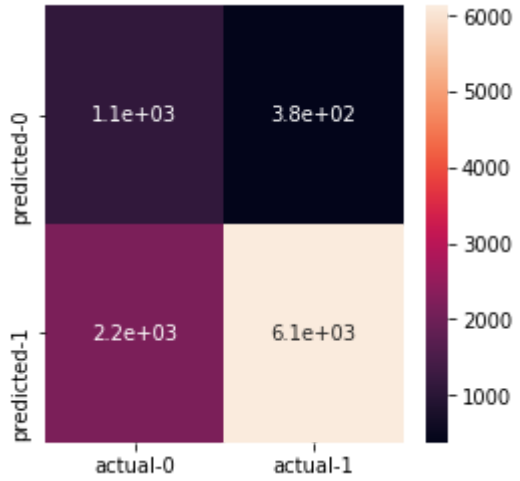
```
====================================================================================
==========================
the maximum value of tpr*(1-fpr) 0.5521904618232139 for threshold 0.884
TRAIN CONFUSION MATRIX
[[1118  376]
 [2177 6129]]
test confusion matrix
[[ 482  433]
 [1453 3632]]
```

In [92]:

```
#printing heatmap for train confusion matrix
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

array=[[1118,376],[2177,6129]]

train=pd.DataFrame(array,index=['predicted-0','predicted-1'],columns=['actual-0','actua
l-1'])
plt.figure(figsize=(4,4))
sn.heatmap(train,annot=True)
```

Out[92]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x201efd639e8>
```



OBSERVATIONS:

Here we plotted the heatmap of confusion matrix of train data.

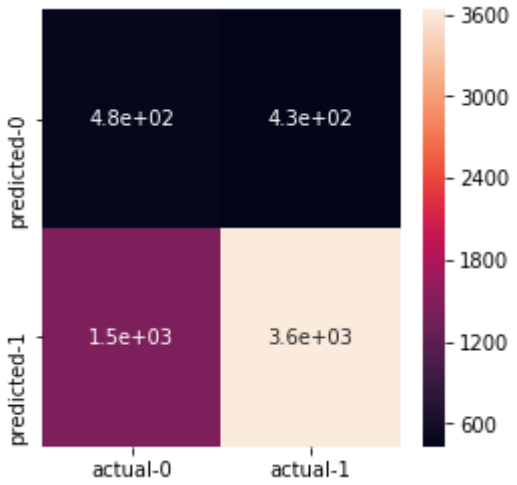
Here the values of tpr and tnr are a bit large which shows the efficiency of the model.

In [93]:

```
#printing the confusion matrix for test set
testarray=[[482,433],[1453,3632]]
test=pd.DataFrame(testarray,index=['predicted-0','predicted-1'],columns=['actual-0','ac
tual-1'])
plt.figure(figsize=(4,4))
sn.heatmap(test,annot=True)
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x201a600d6d8>
```



OBSERVATIONS:

Here we plooted the heatmap of confusion matrix of test data.
The values of tpr and tnr are large compared to fpr and fnr.
This shows the better predictibility of the model.

2.4.1.1 Top 10 important features of positive class from SET 1

In [70]:

```
# Please write all the code with proper documentation
#https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-ba
yes
top_10_positive=naive_bayes.feature_log_prob_[1,0:15].argsort()
print(np.take(vectorizer.get_feature_names(),top_10_positive[:10]))
```

```
['21st century learning' '1st' '21st century learners' '3d' '3d printing'
 '3rd' '1st graders' '21st' '1st grade' '21st century']
```

2.4.1.2 Top 10 important features of negative class from SET 1

In [71]:

```
# Please write all the code with proper documentation
top_10_negative=naive_bayes.feature_log_prob_[0,0:10].argsort()
print(np.take(vectorizer.get_feature_names(),top_10_negative[:10]))
```

```
['21st century learning' '1st' '21st century learners' '1st graders'
 '21st' '21st century' '1st grade' '2016' '2nd' '2017']
```

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [72]:

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

In [73]:

```
#tfidf vectorizing project essay
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer=TfidfVectorizer(min_df=10)
vectorizer.fit(X_train["essay"].values)
train_tfidf=vectorizer.transform(X_train['essay'].values)
cv_tfidf=vectorizer.transform(X_cv['essay'].values)
test_tfidf=vectorizer.transform(X_test['essay'].values)

print("AFTER VECTORIZATION")
print("="*50)
print(train_tfidf.shape,y_train.shape)
print(cv_tfidf.shape,y_cv.shape)
print(test_tfidf.shape,y_test.shape)
```

```
AFTER VECTORIZATION
==================================================
(9800, 6349) (9800,)
(4200, 6349) (4200,)
(6000, 6349) (6000,)
```

In [74]:

```
#vectorizing project title
vectorizer=TfidfVectorizer(min_df=10)
vectorizer.fit(X_train["project_title"].values)
title_train_tfidf=vectorizer.transform(X_train['project_title'].values)
title_cv_tfidf=vectorizer.transform(X_cv['project_title'].values)
title_test_tfidf=vectorizer.transform(X_test['project_title'].values)
print("AFTER VECTORIZATION")
print("="*50)
print(title_train_tfidf.shape,y_train.shape)
print(title_cv_tfidf.shape,y_cv.shape)
print(title_test_tfidf.shape,y_test.shape)
```

```
AFTER VECTORIZATION
==================================================
(9800, 672) (9800,)
(4200, 672) (4200,)
(6000, 672) (6000,)
```

In [94]:

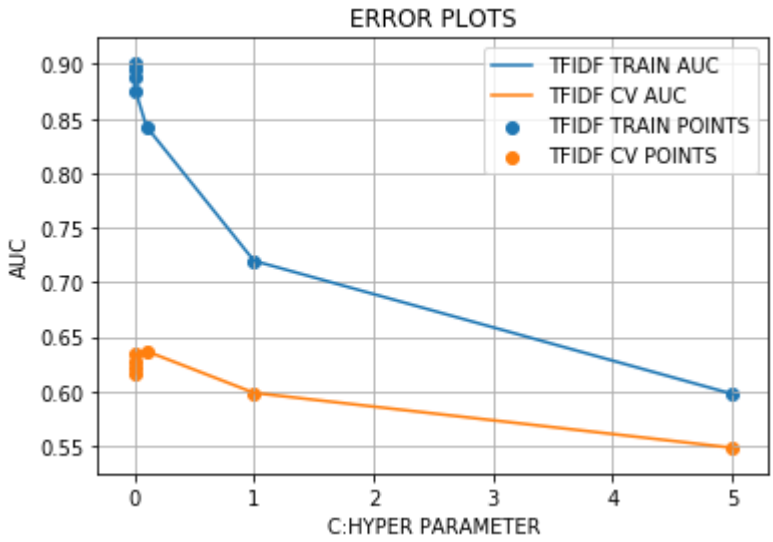
```
#creating final data matrix
from scipy.sparse import hstack
final_train_tfidf=hstack((X_train_state_encoded,X_train_prefix_encoded,X_train_grade_encoded,X_train_categories_encoded,X_train_subcategories_encoded,X_train_price_norm,X_train_projects_norm,X_train_quantity_norm,train_tfidf,title_train_tfidf)).tocsr()
final_cv_tfidf=hstack((X_cv_state_encoded,X_cv_prefix_encoded,X_cv_grade_encoded,X_cv_categories_encoded,X_cv_subcategories_encoded,X_cv_price_norm,X_cv_projects_norm,X_cv_quantity_norm,cv_tfidf,title_cv_tfidf)).tocsr()
final_test_tfidf=hstack((X_test_state_encoded,X_test_prefix_encoded,X_test_grade_encoded,X_test_categories_encoded,X_test_subcategories_encoded,X_test_price_norm,X_test_projects_norm,X_test_quantity_norm,test_tfidf,title_test_tfidf)).tocsr()
print(final_train_tfidf.shape,y_train.shape)
print(final_cv_tfidf.shape,y_cv.shape)
print(final_test_tfidf.shape,y_test.shape)
```

```
(9800, 7124) (9800,)
(4200, 7124) (4200,)
(6000, 7124) (6000,)
```

In [95]:

```
#plotting error plots
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

tfidf_train_auc=[]
tfidf_cv_auc=[]
c=[0.00001,0.0001,0.001,0.01,0.1,1,5]
for i in tqdm(c):
    naive_bayes=MultinomialNB(alpha=i)
    naive_bayes.fit(final_train_tfidf,y_train)

    y_tr_pred=batch_predict(naive_bayes,final_train_tfidf)
    y_cv_pred=batch_predict(naive_bayes,final_cv_tfidf)


    tfidf_train_auc.append(roc_auc_score(y_train,y_tr_pred))
    tfidf_cv_auc.append(roc_auc_score(y_cv,y_cv_pred))

plt.plot(c,tfidf_train_auc,label='TFIDF TRAIN AUC')
plt.plot(c,tfidf_cv_auc,label="TFIDF CV AUC")

plt.scatter(c,tfidf_train_auc,label='TFIDF TRAIN POINTS')
plt.scatter(c,tfidf_cv_auc,label='TFIDF CV POINTS')

plt.legend()
plt.title("ERROR PLOTS")
plt.xlabel("C:HYPER PARAMETER")
plt.ylabel("AUC")
plt.grid()
plt.show()
```

```
100%|██████████| 7/7 [00:01<00:00,  6.18it/s]
```



OBSERVATIONS:

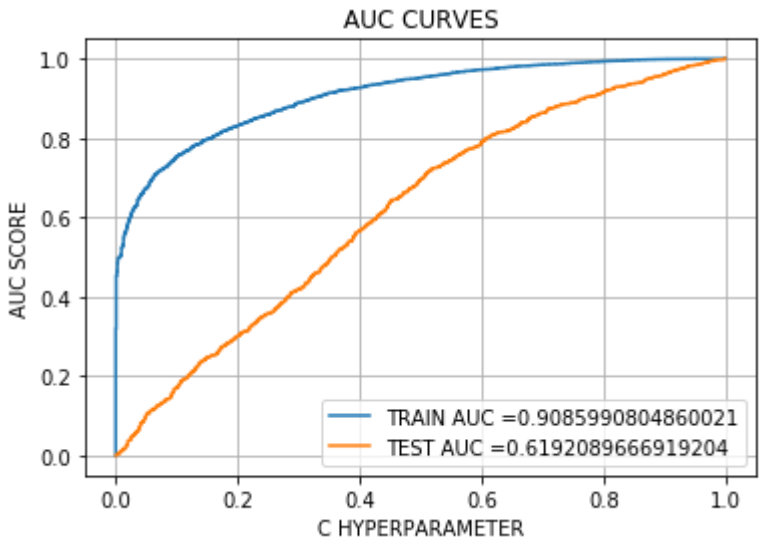
Here we plotted error plots for various values of alpha ranging between 0.0000 1 to 5.

As the difference between train auc and cv auc is similar between for some values we choose our alpha to be 1.

In [96]:

```
#plotting auc curves
from sklearn.metrics import roc_curve,auc

naive_bayes=MultinomialNB(alpha=np.log(1))
naive_bayes.fit(final_train_tfidf,y_train)

y_train_pred=batch_predict(naive_bayes,final_train_tfidf)
y_test_pred=batch_predict(naive_bayes,final_test_tfidf)

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

plt.plot(train_fpr,train_tpr,label="TRAIN AUC ="+str(auc(train_fpr,train_tpr)))
plt.plot(test_fpr,test_tpr,label="TEST AUC ="+str(auc(test_fpr,test_tpr)))

plt.legend()
plt.xlabel("C HYPERPARAMETER")
plt.ylabel('AUC SCORE')
plt.title("AUC CURVES")
plt.grid()
plt.show()
```

```
C:\Users\DELL\Anaconda3\lib\site-packages\sklearn\naive_bayes.py:480: User
Warning:

alpha too small will result in numeric errors, setting alpha = 1.0e-10
```



OBSERVATIONS:

We apply log to best c while testing the model.

By plotting for alpha = log(1) we got a train auc as 90.85% and test auc as 61.92%.

We got better auc score for training data than test data

In [97]:

```
#printing confusion matrix
print('='*100)
from sklearn.metrics import confusion_matrix
best_t=find_best_threshold(tr_threshold,train_fpr,train_tpr)
print('TRAIN CONFUSION MATRIX')
print(confusion_matrix(y_train,predict_with_best_t(y_train_pred,best_t)))
print('test confusion matrix')
print(confusion_matrix(y_test,predict_with_best_t(y_test_pred,best_t)))
```

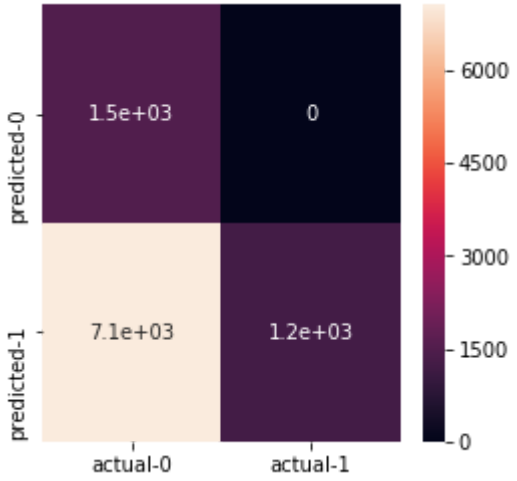
```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.6810497467839091 for threshold 1.0
TRAIN CONFUSION MATRIX
[[1494    0]
 [7057 1249]]
test confusion matrix
[[ 854   61]
 [4468  617]]
```

In [83]:

```
#printing heatmap for train confusion matrix
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

array=[[1494,0],[7057,1249]]

train=pd.DataFrame(array,index=['predicted-0','predicted-1'],columns=['actual-0','actua
l-1'])
plt.figure(figsize=(4,4))
sn.heatmap(train,annot=True)
```

Out[83]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x201a44fd080>
```



OBSERVATIONS:

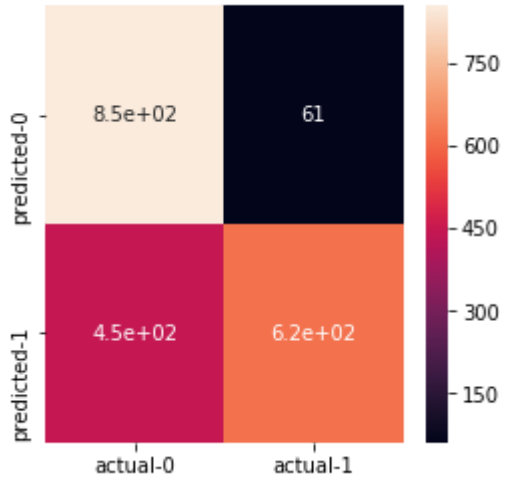
Here we plotted the heatmap of confusion matrix of train data.
we got low fnr but we got a higher value for fpr which is not good.

In [84]:

```
#printing heatmap for test confusion matrix
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

testarray=[[854,61],[448,617]]

test=pd.DataFrame(testarray,index=['predicted-0','predicted-1'],columns=['actual-0','ac
tual-1'])
plt.figure(figsize=(4,4))
sn.heatmap(test,annot=True)
```

Out[84]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x201a52622e8>
```



OBSERVATIONS:

Here we plotted the heatmap of confusion matrix of test data.
Here we got better values for fpr,tpr compared to train data.

2.4.2.1 Top 10 important features of positive class from SET 2

In [85]:

```
# Please write all the code with proper documentation
top_10_positive=naive_bayes.feature_log_prob_[1,0:15].argsort()
print(np.take(vectorizer.get_feature_names(),top_10_positive[:10]))
```

```
['5th' '1st' '4th' 'access' 'action' 'active' '2017' '3d' '2016' '3rd']
```

2.4.2.2 Top 10 important features of negative class from SET 2

In [86]:

```
# Please write all the code with proper documentation
top_10_negative=naive_bayes.feature_log_prob_[0,0:15].argsort()
print(np.take(vectorizer.get_feature_names(),top_10_negative[:15]))
```

```
['5th' '1st' '4th' 'access' 'active' '2017' 'action' '3d' '3rd' '2016'
 '21st' 'academic' 'activities' 'about' '2nd']
```

3. Conclusions

In [87]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable
x=PrettyTable(['vectorizer','best_alpha','train_auc','test_auc'])
x.add_row(['bag of words',1,0.818503,0.665733])
x.add_row(['tfidf',1,0.908599,0.619208])

print(x.get_string(start=0,end=5))
```

```
+--------------+------------+-----------+----------+
|  vectorizer  | best_alpha | train_auc | test_auc |
+--------------+------------+-----------+----------+
| bag of words |     1      |  0.818503 | 0.665733 |
|    tfidf     |     1      |  0.908599 | 0.619208 |
+--------------+------------+-----------+----------+
```

OBSERAVTIONS:

Here we plotted 2 vectorizers for the donor choose data but we only took 20k samples.

In bag of words model there is little less difference between the values of train and test auc scores than compared to tfidf values.

The confusion matrix also has better tnr and tpr values for bag of words vectorizer than compared to tfidf vectorizer.

From pretty table we observe that efficiency is almost 90% for train data in tfidf vectorizer but the differnce between train and test auc is a bit high.