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*

Feature	Description
`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
```

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/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
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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "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('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:

```
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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "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://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 [7]:

```
# 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 [8]:

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

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade

In [9]:

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

In [10]:

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

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with 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 En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
```

develop early reading skills.\r\n\r\nStudents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and ed ucational dvd's for the years to come for other EL students.\r\nnannan
==========
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the bea utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t he hard work put in during the school year, with a dunk tank being the most popular activity.My st udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to hav e an individual 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 us ed by the 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 missing, 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 i n group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta ken. 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 students to do desk work and move at th e same time. These stools will help students 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, these chairs will take away the barrier that exists in schools for a child who can't sit still.nannan
==========
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting 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 races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage r learners; they are like sponges, absorbing all the information and experiences and keep on wanti ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin g 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 pic tures 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 t o make our new school year a very successful one. Thank you!nannan
==========
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th eir 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 pr ice 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 gr oove 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 dev elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l earn through games, my kids don't want to sit and do worksheets. They want to learn to count by ju mping and playing. Physical engagement is the key to our success. The number toss and color and sh ape 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 grea t teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% Af rican-American, making up the largest segment of the student body. A typical school in Dallas is m ade up of 23.2% African-American students. Most of the students are on free or reduced lunch. We a ren'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 smar t, effective, efficient, and disciplined students with good character.In our classroom we can util ize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the so und enough to receive the message. Due to the volume of my speaker my students can't hear videos o r books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my s tudents will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will all ow 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

about different letters and it is more accessible.nannan
==

In [11]:

```
# 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 [12]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir 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 pr
ice 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 gr
oove 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 dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids do not want to sit and do worksheets. They want to learn to count by j
umping and playing. Physical engagement is the key to our success. The number toss and color and s
hape 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 [13]:

```
# \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, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir 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 reduced price
lunch.  Despite their disabilities and limitations, my students love coming to school and come eag
er 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 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 develo
p their core, which enhances gross motor and in Turn fine motor skills.   They also want to learn t
hrough games, my kids do not want to sit and do worksheets. They want to learn to count by jumping
and playing. Physical engagement is the key to our success. The number toss and color and shape ma
ts can make that happen. My students will forget they are doing work and just have the fun a 6 yea
r old deserves.nannan
```

In [14]:

```
#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 cognitiv
e delays gross fine motor delays to autism They are 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 studen
ts 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 lea
rn and explore Have you ever felt like you had ants in your pants and you needed to groove and mov
e as you were in a meeting This is how my kids feel all the time The want to be able to move as th
ey learn or so they say Wobble chairs are the answer and I love then because they develop their co
re which enhances gross motor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Ph
ysical 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 nan
nan
```

In [15]:

```
# 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', 'e
ach', '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', "d
esn'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 [16]:

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

```
100%|██████████████████████████████████████████████████████████████████| 109248/109248
[01:44<00:00, 1044.60it/s]
```

In [17]:

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

Out[17]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gros
s fine motor delays autism they eager beavers always strive work hardest working past limitations
the materials ones i seek students i teach title i school students receive free reduced price lunc
h despite disabilities limitations students love coming school come eager learn explore have ever
felt like ants pants needed groove move meeting this kids feel time the want able move learn say w
obble chairs answer i love develop core enhances gross motor turn fine motor skills they also want
learn games kids not want sit worksheets they want learn count jumping playing physical engagement
key success the number toss color shape mats make happen my students forget work fun 6 year old de
serves nannan'
```

1.4 Preprocessing of `project_title`

In [18]:

```
# similarly you can preprocess the titles also
# Using above lines of code for preprocessing Title text

from tqdm import tqdm
preprocessed_title = []

for sentance in tqdm(project_data['project_title'].values):
    sentTitle = decontracted(sentance)
    sentTitle = sentTitle.replace('\\r', ' ')
    sentTitle = sentTitle.replace('\\"', ' ')
    sentTitle = sentTitle.replace('\\n', ' ')
    sentTitle = re.sub('[^A-Za-z0-9]+', ' ', sentTitle)
    # https://gist.github.com/sebleier/554280
    sentTitle = ' '.join(e for e in sentTitle.split() if e not in stopwords)
    preprocessed_title.append(sentTitle.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:05<00:00, 19850.12it/s]
```

In [21]:

```
# after preprocessing Project title

preprocessed_title[20000]
```

Out[21]:

```
'we need to move it while we input it'
```

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_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [23]:

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

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

In [24]:

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [42]:

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

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

In [26]:

```
# Similarly you can vectorize for title also
# Using above lines of code

vectorizerTitle = CountVectorizer(min_df=10)
text_bow_title = vectorizerTitle.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encodig ",text_bow_title.shape)
```

```
print("Shape of matrix after one hot encodig ",text_bow_title.shape)
```

```
Shape of matrix after one hot encodig  (109248, 3329)
```

1.5.2.2 TFIDF vectorizer

In [28]:

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

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

In [29]:

```
# Similarly you can vectorize for title also
# Using above lines of code

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

```
Shape of matrix after one hot encodig  (109248, 3329)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [30]:

```
'''
# 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-pickle-to-sa
ve-and-load-variables-in-python/

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


'''
```

Out[30]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef
loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\',
encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n
word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.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 tha
t are present in both glove vectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\n
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [31]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-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 [32]:

```
# 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
[01:05<00:00, 1660.80it/s]

109248
300
```

In [33]:

```
# Similarly you can vectorize for title also
# Using above lines of code
# average Word2Vec for Project Title

avg_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_title): # 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_title.append(vector)

print(len(avg_w2v_vectors_title))
print(len(avg_w2v_vectors_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:03<00:00, 33053.54it/s]
```

```
109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [34]:

```
# 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 [35]:

```
# 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((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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
[08:37<00:00, 227.10it/s]
```

```
109248
300
```

In [36]:

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

```
tfidf_model_title = TfidfVectorizer()
tfidf_model_title.fit(preprocessed_title)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary_title = dict(zip(tfidf_model_title.get_feature_names(), list(tfidf_model_title.idf_)))
tfidf_words_title = set(tfidf_model_title.get_feature_names())
```

In [37]:

```
# Using above lines of code
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(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_title):
            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_title[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_title.append(vector)

print(len(tfidf_w2v_vectors_title))
print(len(tfidf_w2v_vectors_title[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:06<00:00, 16554.26it/s]
```

```
109248
300
```

1.5.3 Vectorizing Numerical features

In [38]:

```
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 [39]:

```
# 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 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 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.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

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

In [40]:

```
price_standardized
```

```
price_standardized
```

Out[40]:

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

1.5.4 Merging all the above features

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

In [43]:

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

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

In [44]:

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

Out[44]:

```
(109248, 16663)
```

In [45]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# 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 value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning
3. **Feature importance**
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using

values of `feature_log_prob_` parameter of MultinomialNB and print their corresponding feature names

4. **Representation of results**

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

5. **Conclusion**

- You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

2. Naive Bayes

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

In [46]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# 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

# Data spliting
# Using project_data, which is a merge of project_data , price_date tables

from sklearn.model_selection import train_test_split
Donor_train, Donor_test, Approved_train, Approved_test = train_test_split(project_data, project_dat
a['project_is_approved'], test_size=0.33, stratify=project_data['project_is_approved'])
Donor_train, Donor_cv, Approved_train, Approved_cv = train_test_split(Donor_train, Approved_train,
test_size=0.33, stratify=Approved_train)
```

In [47]:

```
print(Donor_train.shape,Approved_train.shape)
print(Donor_test.shape,Approved_test.shape)
print(Donor_cv.shape,Approved_cv.shape)
project_data.columns
```

```
(49041, 20) (49041,)
(36052, 20) (36052,)
(24155, 20) (24155,)
```

Out[47]:

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

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2 Make Data Model Ready: encoding numerical, categorical features

In [48]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# 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
# One hot encoding the catogorical features : School State, Clean Categories, Clean Sub-Categories
, Project Grade and Teacher Prefix

# One hot Encoding for School State
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['school_state'].values) # fit has to happen only on train data
Donor_train_state_ohe = vectorizer.transform(Donor_train['school_state'].values)
Donor_cv_state_ohe = vectorizer.transform(Donor_cv['school_state'].values)
Donor_test_state_ohe = vectorizer.transform(Donor_test['school_state'].values)

# Print One Hot Encoding - School State output
print("After vectorizations School state")
print(Donor_train_state_ohe.shape, Approved_train.shape)
print(Donor_cv_state_ohe.shape, Approved_cv.shape)
print(Donor_test_state_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations School state
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv
', 'wy']
====================================================================================================
```

In [49]:

```
# Preprocessing Project grade
from collections import Counter
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.split("'",1))

project_grade_category_dict = dict(my_counter)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), key=lambda
kv: kv[1]))
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys()), lowercase
=False, binary=True)

# One Hot Encoding - Project grade category

vectorizer.fit(Donor_train['project_grade_category'].values) # fit has to happen only on train
data
Donor_train_grade_ohe = vectorizer.transform(Donor_train['project_grade_category'].values)
Donor_cv_grade_ohe = vectorizer.transform(Donor_cv['project_grade_category'].values)
Donor_test_grade_ohe = vectorizer.transform(Donor_test['project_grade_category'].values)

# Print One Hot Encoding - Project grade output
print("After vectorizations Project grade category")
print(Donor_train_grade_ohe.shape, Approved_train.shape)
print(Donor_cv_grade_ohe.shape, Approved_cv.shape)
print(Donor_test_grade_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations Project grade category
```

```
(49041, 4) (49041,)
(24155, 4) (24155,)
(36052, 4) (36052,)
['Grades 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2']
==================================================================================================
```

In [50]:

```
# One hot Encoding for project subject categories
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['clean_categories'].values) # fit has to happen only on train data
Donor_train_clean_cat_ohe = vectorizer.transform(Donor_train['clean_categories'].values)
Donor_cv_clean_cat_ohe = vectorizer.transform(Donor_cv['clean_categories'].values)
Donor_test_clean_cat_ohe = vectorizer.transform(Donor_test['clean_categories'].values)

# Print One Hot Encoding - Project subject output
print("After vectorizations project subject categories")
print(Donor_train_clean_cat_ohe.shape, Approved_train.shape)
print(Donor_cv_clean_cat_ohe.shape, Approved_cv.shape)
print(Donor_test_clean_cat_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations project subject categories
(49041, 9) (49041,)
(24155, 9) (24155,)
(36052, 9) (36052,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language',
'math_science', 'music_arts', 'specialneeds', 'warmth']
==================================================================================================
```

In [51]:

```
# One hot Encoding for project subject subcategories
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['clean_subcategories'].values) # fit has to happen only on train data
Donor_train_clean_subcat_ohe = vectorizer.transform(Donor_train['clean_subcategories'].values)
Donor_cv_clean_subcat_ohe = vectorizer.transform(Donor_cv['clean_subcategories'].values)
Donor_test_clean_subcat_ohe = vectorizer.transform(Donor_test['clean_subcategories'].values)

# Print One Hot Encoding - project subject subcategories output
print("After vectorizations project subject subcategories")
print(Donor_train_clean_subcat_ohe.shape, Approved_train.shape)
print(Donor_cv_clean_subcat_ohe.shape, Approved_cv.shape)
print(Donor_test_clean_subcat_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations project subject subcategories
(49041, 30) (49041,)
(24155, 30) (24155,)
(36052, 30) (36052,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government',
'college_careerprep', 'communityservice', 'earlydevelopment', 'economics', 'environmentalscience',
'esl', 'extracurricular', 'financialliteracy', 'foreignlanguages', 'gym_fitness',
'health_lifescience', 'health_wellness', 'history_geography', 'literacy', 'literature_writing', 'm
athematics', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'performingarts', 'socia
lsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
==================================================================================================
```

In [53]:

```
# To avoid np.NaN invalid document error;
# Source : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-
valueerror-np-nan-is-an-invalid-document/39308809
# One hot Encoding for Teacher Prefix
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['teacher_prefix'].values.astype('U')) # fit has to happen only on train
data
Donor_train_teacher_ohe = vectorizer.transform(Donor_train['teacher_prefix'].values.astype('U'))
```

```
Donor_cv_teacher_ohe = vectorizer.transform(Donor_cv['teacher_prefix'].values.astype('U'))
Donor_test_teacher_ohe = vectorizer.transform(Donor_test['teacher_prefix'].values.astype('U'))

# Print One Hot Encoding - Teacher Prefix output
print("After vectorizations Teacher Prefix")
print(Donor_train_teacher_ohe.shape, Approved_train.shape)
print(Donor_cv_teacher_ohe.shape, Approved_cv.shape)
print(Donor_test_teacher_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations Teacher Prefix
(49041, 6) (49041,)
(24155, 6) (24155,)
(36052, 6) (36052,)
['dr', 'mr', 'mrs', 'ms', 'nan', 'teacher']
====================================================================================================
```

Numerical Features

In [54]:

```
# Normalizing the Numerical data : Price
# Using code from Sample solution

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(Donor_train['price'].values.reshape(-1,1))
Donor_train_price_norm = normalizer.transform(Donor_train['price'].values.reshape(-1,1))
Donor_cv_price_norm = normalizer.transform(Donor_cv['price'].values.reshape(-1,1))
Donor_test_price_norm = normalizer.transform(Donor_test['price'].values.reshape(-1,1))

print("After vectorizations Numerical Data: Price")
print(Donor_train_price_norm.shape, Approved_train.shape)
print(Donor_cv_price_norm.shape, Approved_cv.shape)
print(Donor_test_price_norm.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizations Numerical Data: Price
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

In [55]:

```
# Normalizing the Numerical data : teacher_number_of_previously_posted_projects

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(Donor_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
Donor_train_postedCount_norm =
normalizer.transform(Donor_train['teacher_number_of_previously_posted_projects'].values.reshape(-1
,1))
Donor_cv_postedCount_norm =
normalizer.transform(Donor_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

Donor_test_postedCount_norm =
normalizer.transform(Donor_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,
1))

print("After vectorizations Numerical Data: Previously Posted Projects")
print(Donor_train_postedCount_norm.shape, Approved_train.shape)
print(Donor_cv_postedCount_norm.shape, Approved_cv.shape)
print(Donor_test_postedCount_norm.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizations Numerical Data: Previously Posted Projects
(49041, 1) (49041,)
```

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

In [56]:

```
# Normalizing the Numerical data : Quantity
# Using code from Sample solution

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(Donor_train['quantity'].values.reshape(-1,1))
Donor_train_quantity_norm = normalizer.transform(Donor_train['quantity'].values.reshape(-1,1))
Donor_cv_quantity_norm = normalizer.transform(Donor_cv['quantity'].values.reshape(-1,1))
Donor_test_quantity_norm = normalizer.transform(Donor_test['quantity'].values.reshape(-1,1))

print("After vectorizations Numerical Data: Quantity")
print(Donor_train_quantity_norm.shape, Approved_train.shape)
print(Donor_cv_quantity_norm.shape, Approved_cv.shape)
print(Donor_test_quantity_norm.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizations Numerical Data: Quantity
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

2.3 Make Data Model Ready: encoding eassay, and project_title

BoW : Project Essays

In [57]:

```
# Using sample solution code
# Using max features as 5000, to make column size same

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(Donor_train['essay'].values) # fit has to happen only on train data
Donor_train_essay_bow = vectorizer.transform(Donor_train['essay'].values)
Donor_cv_essay_bow = vectorizer.transform(Donor_cv['essay'].values)
Donor_test_essay_bow = vectorizer.transform(Donor_test['essay'].values)

print("After vectorizing Project Essays BoW")
print(Donor_train_essay_bow.shape, Approved_train.shape)
print(Donor_cv_essay_bow.shape, Approved_cv.shape)
print(Donor_test_essay_bow.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizing Project Essays BoW
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================
```

BoW : Project Title

In [58]:

```
# Using sample solution code
# Using max features as 3000, to make column size same

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=3000)
```

```
vectorizer.fit(Donor_train['project_title'].values) # fit has to happen only on train data
Donor_train_title_bow = vectorizer.transform(Donor_train['project_title'].values)
Donor_cv_title_bow = vectorizer.transform(Donor_cv['project_title'].values)
Donor_test_title_bow = vectorizer.transform(Donor_test['project_title'].values)

print("After vectorizing Project Essays BoW")
print(Donor_train_title_bow.shape, Approved_train.shape)
print(Donor_cv_title_bow.shape, Approved_cv.shape)
print(Donor_test_title_bow.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizing Project Essays BoW
(49041, 3000) (49041,)
(24155, 3000) (24155,)
(36052, 3000) (36052,)
====================================================================================================
```

TFIDF : Project Essays

In [75]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(Donor_train['essay'].values) # fit has to happen only on train data
Donor_train_essay_tfidf = vectorizer.transform(Donor_train['essay'].values)
Donor_cv_essay_tfidf = vectorizer.transform(Donor_cv['essay'].values)
Donor_test_essay_tfidf = vectorizer.transform(Donor_test['essay'].values)

print("After vectorizing Project Essays TFIDF")
print(Donor_train_essay_tfidf.shape, Approved_train.shape)
print(Donor_cv_essay_tfidf.shape, Approved_cv.shape)
print(Donor_test_essay_tfidf.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizing Project Essays TFIDF
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================
```

TFIDF : Project Title

In [78]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=3000)
vectorizer.fit(Donor_train['project_title'].values) # fit has to happen only on train data
Donor_train_title_tfidf = vectorizer.transform(Donor_train['project_title'].values)
Donor_cv_title_tfidf = vectorizer.transform(Donor_cv['project_title'].values)
Donor_test_title_tfidf = vectorizer.transform(Donor_test['project_title'].values)

print("After vectorizing Project Title using TFIDF: ")
print(Donor_train_title_tfidf.shape, Approved_train.shape)
print(Donor_cv_title_tfidf.shape, Approved_cv.shape)
print(Donor_test_title_tfidf.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizing Project Title using TFIDF:
(49041, 3000) (49041,)
(24155, 3000) (24155,)
(36052, 3000) (36052,)
====================================================================================================
```

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

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)

In [59]:

```
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
# Using sample soultion code

from scipy.sparse import hstack
Donor_tr = hstack((Donor_train_essay_bow,Donor_train_title_bow, Donor_train_state_ohe,
Donor_train_teacher_ohe, Donor_train_grade_ohe,
Donor_train_clean_cat_ohe,Donor_train_clean_subcat_ohe,Donor_train_price_norm,Donor_train_postedCou
nt_norm,Donor_train_quantity_norm)).tocsr()
Donor_cr = hstack((Donor_cv_essay_bow,Donor_cv_title_bow, Donor_cv_state_ohe, Donor_cv_teacher_ohe
, Donor_cv_grade_ohe,
Donor_cv_clean_cat_ohe,Donor_cv_clean_subcat_ohe,Donor_cv_price_norm,Donor_cv_postedCount_norm,Don
or_cv_quantity_norm)).tocsr()
Donor_te = hstack((Donor_test_essay_bow,Donor_test_title_bow, Donor_test_state_ohe,
Donor_test_teacher_ohe, Donor_test_grade_ohe, Donor_test_clean_cat_ohe,Donor_test_clean_subcat_ohe
,Donor_test_price_norm,Donor_test_postedCount_norm,Donor_test_quantity_norm)).tocsr()

print("Final Donor Data Matrix for Set 1")
print(Donor_tr.shape,Approved_train.shape)
print(Donor_cr.shape,Approved_cv.shape)
print(Donor_te.shape,Approved_test.shape)
print("="*100)
```

```
Final Donor Data Matrix for Set 1
(49041, 8103) (49041,)
(24155, 8103) (24155,)
(36052, 8103) (36052,)
====================================================================================================
```

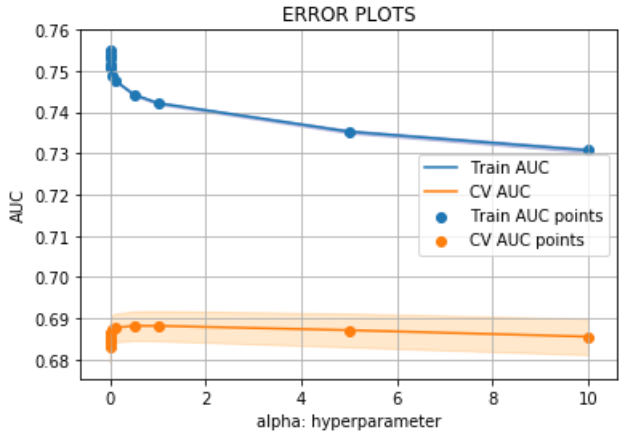
In [60]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [62]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB()
parameters = {'alpha':[10,5,1,0.5,0.1,0.05,0.01,0.005,0.001,0.0005,0.0001]}
clf = GridSearchCV(nb, parameters, cv=3, scoring='roc_auc')
clf.fit(Donor_tr, Approved_train)
```

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

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

#source: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')

#source: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color=
'darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

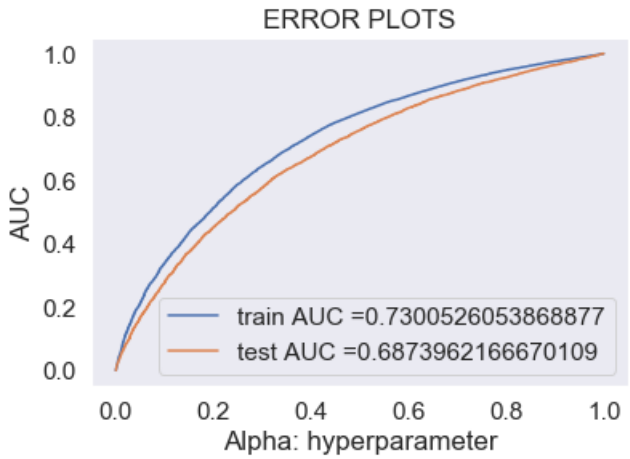



```
Wall time: 16.2 s
```

In [74]:

```
%%time
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.naive_bayes import MultinomialNB


nb = MultinomialNB(alpha=0.1)
nb.fit(Donor_tr, Approved_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

Approved_train_pred = batch_predict(nb, Donor_tr)
Approved_test_pred = batch_predict(nb, Donor_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(Approved_train, Approved_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(Approved_test, Approved_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()
```




```
Wall time: 1.7 s
```

In [69]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[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.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

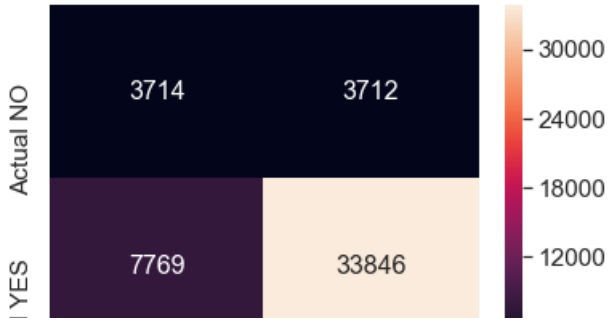
In [71]:

```
## Confustion Matrix for Set 1 Train vs Test data

print("Train confusion matrix for Set 1")
donor_SET1_tr = pd.DataFrame(confusion_matrix(Approved_train, predict(Approved_train_pred, tr_thres
holds, train_fpr, train_fpr)))
donor_SET1_tr.columns = ['Predicted NO','Predicted YES']
donor_SET1_tr = donor_SET1_tr.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)#for label size
sns.heatmap(donor_SET1_tr, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train confusion matrix for Set 1
the maximum value of tpr*(1-fpr) 0.2499999818661462 for threshold 0.064
```

Out[71]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cf2d691fd0>
```



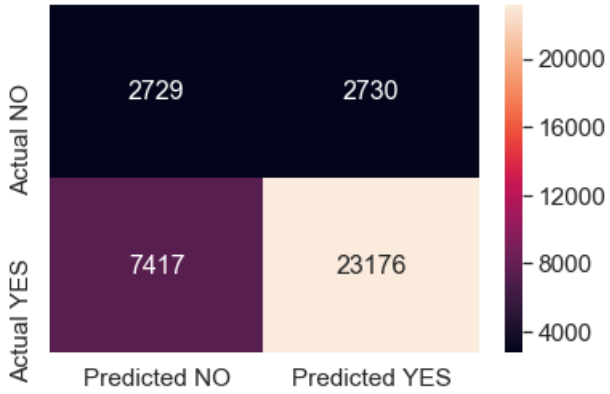
In [72]:

```
print("="*100)
print("Test confusion matrix for Set 1")
donor_SET1_te = pd.DataFrame(confusion_matrix(Approved_test, predict(Approved_test_pred,
te_thresholds, test_fpr, test_fpr)))
donor_SET1_te.columns = ['Predicted NO','Predicted YES']
donor_SET1_te = donor_SET1_te.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)#for label size
sns.heatmap(donor_SET1_te, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================================

Test confusion matrix for Set 1
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.225
```

Out[72]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cf3176a898>
```



2.4.1.1 Top 10 important features of positive and Negative classes from SET 1

In [76]:

```
from prettytable import PrettyTable

def most_informative_feature(vectorizer, classifier, n=10):
    class_labels = classifier.classes_
    feature_names = vectorizer.get_feature_names()
    topn_class1 = sorted(zip(classifier.feature_log_prob_[0], feature_names))[:n]
    topn_class2 = sorted(zip(classifier.feature_log_prob_[0], feature_names))[-n:]
    top = zip(topn_class1, topn_class2)
    x = PrettyTable()
    x.field_names = ["N-value","Negatives","P-value","Positives"]

    for (feature_log_prob_1, fn_1), (feature_log_prob_2, fn_2) in top:
        x.add_row([feature_log_prob_1, fn_1, feature_log_prob_2, fn_2])
    print(x)
```

In [77]:

```
most_informative_feature(vectorizer, nb)
```

```
+--------------------+------------------+--------------------+-----------+
|       N-value      |     Negatives    |       P-value      | Positives |
+--------------------+------------------+--------------------+-----------+
|  -12.23307549010364 |    these stools   |  -4.746303869568903 |   their   |
| -12.159512922926623 |    stools will    |  -4.607737386378522 |    they   |
| -11.665855102782999 |   wobble stools   |  -4.575450038051642 |    are    |
| -11.469918143971046 |  stability balls  |  -4.481876004259757 |     my    |
```

```
| -11.469918143971046 | these chromebooks | -4.294145297700125 |     in    |
|  -11.4349495461333  |    calculators    |  -4.229942154581785 |     of    |
| -11.247486589838926 |   to move while   | -3.9805816537703667 |  students |
| -11.165473438178092 |    hokki stools   |  -3.775695513142624 |    the    |
| -11.139565253319427 |   chromebooks to  |  -3.658289119320793 |    and    |
| -11.139565253319427 |      ipads to     |  -3.501656561074025 |     to    |
+---------------------+-------------------+---------------------+-----------+
```

2.4.2 Applying Naive Bayes on TFIDF, SET 2

Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)

In [79]:

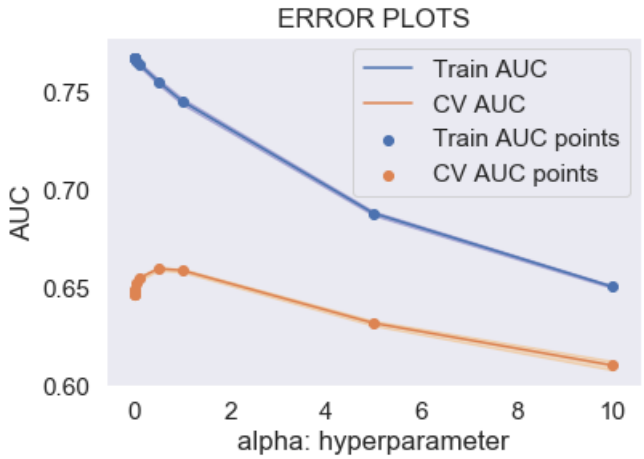
```
# Using sample soultion code

from scipy.sparse import hstack
Donor_tr_tfidf = hstack((Donor_train_essay_tfidf,Donor_train_title_tfidf, Donor_train_state_ohe,
Donor_train_teacher_ohe, Donor_train_grade_ohe,
Donor_train_clean_cat_ohe,Donor_train_clean_subcat_ohe,Donor_train_price_norm,Donor_train_postedCou
nt_norm,Donor_train_quantity_norm)).tocsr()
Donor_cr_tfidf = hstack((Donor_cv_essay_tfidf,Donor_cv_title_tfidf, Donor_cv_state_ohe,
Donor_cv_teacher_ohe, Donor_cv_grade_ohe,
Donor_cv_clean_cat_ohe,Donor_cv_clean_subcat_ohe,Donor_cv_price_norm,Donor_cv_postedCount_norm,Don
or_cv_quantity_norm)).tocsr()
Donor_te_tfidf = hstack((Donor_test_essay_tfidf,Donor_test_title_tfidf, Donor_test_state_ohe,
Donor_test_teacher_ohe, Donor_test_grade_ohe, Donor_test_clean_cat_ohe,Donor_test_clean_subcat_ohe
,Donor_test_price_norm,Donor_test_postedCount_norm,Donor_test_quantity_norm)).tocsr()

print("Final Donor Data Matrix for Set 2")
print(Donor_tr_tfidf.shape,Approved_train.shape)
print(Donor_cr_tfidf.shape,Approved_cv.shape)
print(Donor_te_tfidf.shape,Approved_test.shape)
print("="*100)
```

```
Final Donor Data Matrix for Set 2
(49041, 8103) (49041,)
(24155, 8103) (24155,)
(36052, 8103) (36052,)
====================================================================================================
```

In [80]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB

nb_set2 = MultinomialNB()
parameters = {'alpha':[10,5,1,0.5,0.1,0.05,0.01,0.005,0.001,0.0005,0.0001]}
clf = GridSearchCV(nb_set2, parameters, cv=3, scoring='roc_auc')
clf.fit(Donor_tr_tfidf, Approved_train)

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

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

#source: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')

#source: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color=
'darkorange')

plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
```

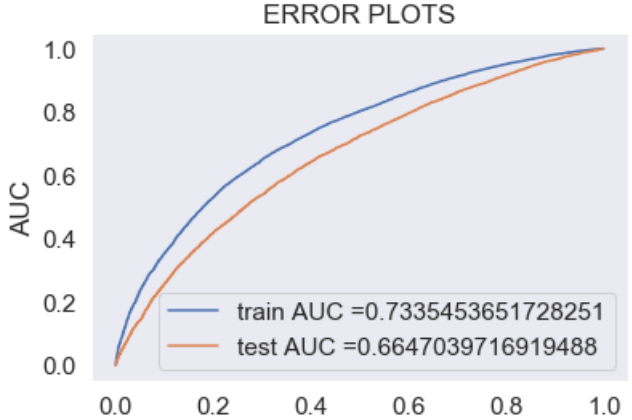
```
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```




```
Wall time: 17.2 s
```

In [88]:

```
%%time
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.naive_bayes import MultinomialNB


nb_set2 = MultinomialNB(alpha=0.5)
nb_set2.fit(Donor_tr_tfidf, Approved_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

Approved_train_pred_tfidf = batch_predict(nb_set2, Donor_tr_tfidf)
Approved_test_pred_tfidf = batch_predict(nb_set2, Donor_te_tfidf)

train_fpr, train_tpr, tr_thresholds = roc_curve(Approved_train, Approved_train_pred_tfidf)
test_fpr, test_tpr, te_thresholds = roc_curve(Approved_test, Approved_test_pred_tfidf)

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()
```



Alpha: hyperparameter

```
Wall time: 911 ms
```

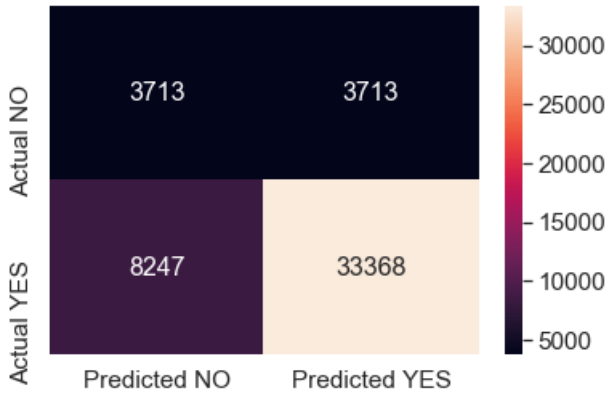
In [89]:

```
## Confustion Matrix for Set 2 Train vs Test data

print("Train confusion matrix for Set 2")
donor_SET2_tr = pd.DataFrame(confusion_matrix(Approved_train, predict(Approved_train_pred_tfidf,
tr_thresholds, train_fpr, train_fpr)))
donor_SET2_tr.columns = ['Predicted NO','Predicted YES']
donor_SET2_tr = donor_SET2_tr.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)#for label size
sns.heatmap(donor_SET2_tr, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train confusion matrix for Set 2
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.78
```

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cf350d15f8>
```



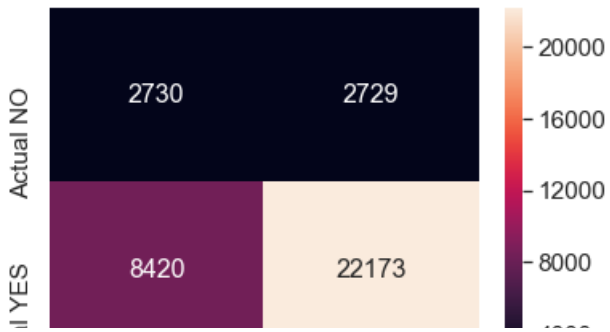
In [91]:

```
## Confustion Matrix for Set 2 Train vs Test data

print("Test confusion matrix for Set 2")
donor_SET2_te = pd.DataFrame(confusion_matrix(Approved_test, predict(Approved_test_pred_tfidf,
te_thresholds, test_fpr, test_fpr)))
donor_SET2_te.columns = ['Predicted NO','Predicted YES']
donor_SET2_te = donor_SET2_te.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)#for label size
sns.heatmap(donor_SET2_te, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test confusion matrix for Set 2
the maximum value of tpr*(1-fpr) 0.24999999161092995 for threshold 0.813
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cf3173d2b0>
```



2.4.2.1 Top 10 important features of positive and negative class from SET 2

In [92]:

```
most_informative_feature(vectorizer, nb_set2)
```

```
+---------------------+----------------+---------------------+------------------+
|       N-value       |   Negatives    |       P-value       |    Positives     |
+---------------------+----------------+---------------------+------------------+
| -11.032407548211268 |  explore the   |  -6.399406685878826 |      leader      |
|  -11.03076368488057 | need your help |  -6.394755051541892 |      member      |
| -10.925915431404311 |    fired up    |  -6.390120088219138 |     powerful     |
|  -10.91356463233264 |      food      |  -6.383424042592046 |    the key to    |
| -10.902254651688603 |   need your    | -6.2920340508697254 | wiggle and learn |
| -10.866146703814168 |     power      |  -6.09019243490633  |    capturing     |
| -10.794830516689192 |    food for    |  -5.957872120053431 |      texts       |
|  -10.78543105181453 |     enjoy      |  -5.840965401756177 |  of technology   |
| -10.768424301426945 |    culture     |  -5.778366983384115 |     towards      |
| -10.684564856664522 |     techy      |  -5.215291672590688 |      autism      |
+---------------------+----------------+---------------------+------------------+
```

3. Conclusions

In [93]:

```
# Please compare all your models using Prettytable library

from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "K-fold","Hyper Parameter", "Train AUC","Test AUC"]
x.add_row(["Set 1", "Naive Bayes", 3, 0.1, 73.0 , 68.7])
x.add_row(["Set 2", "Naive Bayes", 3, 0.5, 73.3 , 66.4])
print(x)
```

```
+------------+-------------+--------+-----------------+-----------+----------+
| Vectorizer |    Model    | K-fold | Hyper Parameter | Train AUC | Test AUC |
+------------+-------------+--------+-----------------+-----------+----------+
|   Set 1    | Naive Bayes |   3    |       0.1       |    73.0   |   68.7   |
|   Set 2    | Naive Bayes |   3    |       0.5       |    73.3   |   66.4   |
+------------+-------------+--------+-----------------+-----------+----------+
```