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`

Feature	**Description** An explanation of the resources needed for the project. **Example:**
`project_resource_summary`	- `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

from 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

2.1 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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # 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]))
```

2.2 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 & Hunger"]
```

```
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=>
"Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e r
emoving '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]))
```

2.3 Text preprocessing of essay

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
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

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(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. We are
a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our s
chool. \r\n\r\n We have over 24 languages represented in our English Learner program with students at e
very 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, bel
iefs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Ou
r English learner's have a strong support system at home that begs for more resources.  Many times our
parents are learning to read and speak English along side of their children.  Sometimes this creates ba
rriers for parents to be able to help their child learn phonetics, letter recognition, and other readin
g skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of t
he English language even if no one at home is able to assist.  All families with students within the Le
vel 1 proficiency status, will be a offered to be a part of this program.  These educational videos wil
l be specially chosen by the English Learner Teacher and will be sent home regularly to watch.  The vid
eos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd
player will have the opportunity to check out a dvd player to use for the year.  The plan is to use the
se videos and educational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love 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% are minority students. \r\nThe school has a vibrant community that loves to get togethe
r and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes tha
t students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, an
d games. At the end of the year the school hosts a carnival to celebrate the hard work put in during th
e school year, with a dunk tank being the most popular activity.My students will use these five brightl
y colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of
ten in the classroom and not enough for each student to have an 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 readin
g times. The rest of the day they will be used 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. Wh
en the students are sitting in group with me on the Hokki Stools, they are always moving, but at the sa
me time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the
first to be taken. There are always students who head over to the kidney table to get one of the stools
who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hou
rs a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the
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 t
ake 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 desk
s, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to c
reate a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class i
s 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 and reduced-price lunch to qualify. Our s
chool is an \"open classroom\" concept, which is very unique as there are no walls separating the class
rooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all t
he information and experiences and keep on wanting more.With these resources such as the comfy red thro
w pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help creat
e the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom en
vironment is very important in the success in each and every child's education. The nautical photo prop
s will be used with each child as they step foot into our classroom for the first time on Meet the Teac
her evening. I'll take pictures of each child with them, have them developed, and then hung in our clas
sroom ready for their first day of 4th grade.  This kind gesture will set the tone before even the firs
t 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 o
ur classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of m
y 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 de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t 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 explo
re.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 s
it and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the ke
```

```
y to our success. The number toss and color and shape mats can make that happen. My students will forge
t 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 tea
cher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-Amer
ican, 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 docto
rs, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspirin
g minds of young children and we focus not only on academics but one smart, effective, efficient, and d
isciplined students with good character.In our classroom we can utilize the Bluetooth for swift transit
ions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due t
o the volume of my speaker my students can't hear videos or books clearly and it isn't making the lesso
ns as meaningful. But with the bluetooth 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 n
eeded for the day and has an extra part to it I can use.  The table top chart has all of the letter, wo
rds and pictures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [11]:

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

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

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

In [12]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t 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 explo
re.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 do not want to
sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the k
ey to our success. The number toss and color and shape mats can make that happen. My students will forg
et they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
```

In [13]:

```python
# \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 de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t 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 thei
r disabilities and limitations, my students love coming to school and come eager to learn and explore.H
```

```
ave you ever felt like you had ants in your pants and you needed to groove and move as you were in a me
eting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.
Wobble chairs are the answer and I love then because they develop their core, which enhances gross moto
r and in Turn fine motor skills.   They also want to learn through games, my kids do not want to sit an
d 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 the
y are doing work and just have the fun a 6 year 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 cognitive del
ays gross fine motor delays to autism They are eager beavers and always strive to work their hardest wo
rking 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 disabilitie
s and limitations my students love coming to school and come eager to learn and explore Have you ever f
elt like you had ants in your pants and you needed to groove and move as you were in a 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 ar
e the answer and I love then because they develop their core which enhances gross motor and in Turn fin
e 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 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 [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', 't
heir',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these',
'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'd
o', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'whil
e', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'bef
ore', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'a
gain', '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 [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 [01:24<00:
00, 1296.69it/s]
```

In [17]:

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

Out[17]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross fin
e motor delays autism they eager beavers always strive work hardest working past limitations the materi
als ones i seek students i teach title i school students receive free reduced price lunch despite disab
ilities 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 wobble chairs answer i love
develop core enhances gross motor turn fine motor skills they also want learn games kids not want sit w
orksheets they want learn count jumping playing physical engagement key success the number toss color s
hape mats make happen my students forget work fun 6 year old deserves nannan'
```

In [18]:

```
project_data['clean_essay'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
project_data.head(2)
```

Out[18]:

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

2.4 Preprocessing of `project_title`

In [19]:

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

In [20]:

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

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [21]:

```
# 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 [22]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar

# https://gist.github.com/sebleier/554280

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 = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████| 109248/109248 [00:03<00:0
0, 30500.95it/s]
```

In [23]:

```
# after preprocesing
preprocessed_titles[20000]
```

Out[23]:

```
'need move input'
```

In [24]:

```
project_data['clean_project_title'] = preprocessed_titles
project_data.drop(['project_title'], axis=1, inplace=True)
project_data.head(2)
```

Out[24]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime

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

2.5 Cleaning data of project_grade_category

In [25]:

```
#cleaning project_grade_category

grades = list(project_data['project_grade_category'].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
grade_list = []
for i in grades:
    i = i.replace('-','_')
    i = i.replace(' ','')

    grade_list.append(i)
```

In [26]:

```
project_data['clean_grade_category'] = grade_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)
```

Out[26]:

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

2.6 Droping unnecessary columns

In [27]:

```
#project_data.drop(['id'], axis=1, inplace=True)
project_data.drop(['teacher_id'], axis=1, inplace=True)
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
project_data.drop(['project_resource_summary'], axis=1, inplace=True)
project_data.drop(['Unnamed: 0'], axis=1, inplace=True)
project_data.head(2)
```

Out[27]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

2.7 Adding price column in our dataframe

In [28]:

```
resource_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1541272 entries, 0 to 1541271
Data columns (total 4 columns):
id             1541272 non-null object
description    1540980 non-null object
quantity       1541272 non-null int64
price          1541272 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 47.0+ MB
```

In [29]:

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

Out[29]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

In [30]:

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

In [31]:

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

Out[31]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

2.8 Preprocessing of teacher_prefix

In [32]:

```
import re
prefix = list(project_data['teacher_prefix'].values)

prefix_list = []

for i in prefix:

    j=str(i)
    j=j.lower()
    j = re.sub(r"\.", "",j)

    prefix_list.append(j)


#print(prefix_list)
```

In [33]:

```
project_data['clean_teacher_prefix'] = prefix_list
project_data.drop(['teacher_prefix'], axis=1, inplace=True)
project_data.head(2)
```

Out[33]:

	id	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projects	project_is_a
0	p253737	IN	2016-12-05 13:43:57	0	0
1	p258326	FL	2016-10-25 09:22:10	7	1

2.9 Preprocessing of school_state

In [34]:

```
state = list(project_data['school_state'].values)

state_list = []

for i in state:

    j=str(i)
    j=j.lower()


    state_list.append(j)
```

```
#print(state_list)
```

In [35]:

```
project_data['clean_school_state'] = state_list
#project_data.drop(['school_state'], axis=1, inplace=True)
project_data.head(2)
```

Out[35]:

	id	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projects	project_is_a
0	p253737	IN	2016-12-05 13:43:57	0	0
1	p258326	FL	2016-10-25 09:22:10	7	1

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

---

3. Naive Bayes

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

In [36]:

```
# 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


from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection


X = project_data.drop(['project_is_approved','id'], axis=1)
X.head(2)

y = project_data['project_is_approved'].values


# split the data set into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,shuffle=True)

# split the train data set into cross validation train and cross validation test
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.2,shuffle=True)

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

```
(69918, 11) (69918,)
(17480, 11) (17480,)
(21850, 11) (21850,)
```

In [37]:

```
X.head(2)
```

Out[37]:

	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projects	clean_categories	clea
0	IN	2016-12-05 13:43:57	0	Literacy_Language	ESL
1	FL	2016-10-25 09:22:10	7	History_Civics Health_Sports	Civi Tea

3.2 Make Data Model Ready: encoding numerical, categorical features

3.2 Make Data Model Ready: encoding numerical, categorical features

In [38]:

```
# 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
```

3.2.1 encoding categorical features: School State

In [39]:

```
vectorizer1 = CountVectorizer()
vectorizer1.fit(X_train['clean_school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer1.transform(X_train['clean_school_state'].values)
X_cv_state_ohe = vectorizer1.transform(X_cv['clean_school_state'].values)
X_test_state_ohe = vectorizer1.transform(X_test['clean_school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer1.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     bow_features.append(i)
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 51) (69918,)
(17480, 51) (17480,)
(21850, 51) (21850,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks',
'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', '
ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
====================================================================================================
```

3.2.2 encoding categorical features: teacher_prefix

In [40]:

```
vectorizer2 = CountVectorizer()
vectorizer2.fit(X_train['clean_teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer2.transform(X_train['clean_teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer2.transform(X_cv['clean_teacher_prefix'].values)
X_test_teacher_ohe = vectorizer2.transform(X_test['clean_teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer2.get_feature_names())
print("="*100)


# for i in vectorizer.get_feature_names():
#     bow_features.append(i)
#     tfidf_features.append(i)
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 6) (69918,)
(17480, 6) (17480,)
(21850, 6) (21850,)
['dr', 'mr', 'mrs', 'ms', 'nan', 'teacher']
====================================================================================================
```

3.2.3 encoding categorical features: project_grade_category

In [41]:

```
vectorizer3 = CountVectorizer()
vectorizer3.fit(X_train['clean_grade_category'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer3.transform(X_train['clean_grade_category'].values)
X_cv_grade_ohe = vectorizer3.transform(X_cv['clean_grade_category'].values)
X_test_grade_ohe = vectorizer3.transform(X_test['clean_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer3.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     bow_features.append(i)
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 4) (69918,)
(17480, 4) (17480,)
(21850, 4) (21850,)
['grades3_5', 'grades6_8', 'grades9_12', 'gradesprek_2']
====================================================================================================
```

3.2.4 encoding categorical features: project_subject_categories

In [42]:

```
vectorizer4 = CountVectorizer()
vectorizer4.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_categories_ohe = vectorizer4.transform(X_train['clean_categories'].values)
X_cv_categories_ohe = vectorizer4.transform(X_cv['clean_categories'].values)
X_test_categories_ohe = vectorizer4.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_categories_ohe.shape, y_train.shape)
print(X_cv_categories_ohe.shape, y_cv.shape)
print(X_test_categories_ohe.shape, y_test.shape)
print(vectorizer4.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     bow_features.append(i)
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 9) (69918,)
(17480, 9) (17480,)
(21850, 9) (21850,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_scienc
e', 'music_arts', 'specialneeds', 'warmth']
```

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

3.2.5 encoding categorical features: project_subject_subcategories

In [43]:

```
vectorizer5 = CountVectorizer()
vectorizer5.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_subcategories_ohe = vectorizer5.transform(X_train['clean_subcategories'].values)
X_cv_subcategories_ohe = vectorizer5.transform(X_cv['clean_subcategories'].values)
X_test_subcategories_ohe = vectorizer5.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_subcategories_ohe.shape, y_train.shape)
print(X_cv_subcategories_ohe.shape, y_cv.shape)
print(X_test_subcategories_ohe.shape, y_test.shape)
print(vectorizer5.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     bow_features.append(i)
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 30) (69918,)
(17480, 30) (17480,)
(21850, 30) (21850,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_careerprep', 'co
mmunityservice', 'earlydevelopment', 'economics', 'environmentalscience', 'esl', 'extracurricular', 'fi
nancialliteracy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wellness', 'history_
geography', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutritioneducation', 'other', 'p
arentinvolvement', 'performingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'war
mth']
====================================================================================================
```

3.2.6 encoding numerical feature: price

In [44]:

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


X_train_price_norm = X_train_price_norm.reshape(-1,1)
X_cv_price_norm = X_cv_price_norm.reshape(-1,1)
X_test_price_norm = X_test_price_norm.reshape(-1,1)


print("After vectorizations")
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)
print("="*100)

# for i in X_train['price']:
#     bow_features.append(i)
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 1) (69918,)
(17480, 1) (17480,)
(21850, 1) (21850,)
====================================================================================================
```

3.2.7 encoding numerical feature: teacher_number_of_previously_posted_projects

In [45]:

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

X_train_posted_project_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_project
s'].values.reshape(1,-1))
X_cv_posted_project_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].va
lues.reshape(1,-1))
X_test_posted_project_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'
].values.reshape(1,-1))

X_train_posted_project_norm = X_train_posted_project_norm.reshape(-1,1)
X_cv_posted_project_norm = X_cv_posted_project_norm.reshape(-1,1)
X_test_posted_project_norm = X_test_posted_project_norm.reshape(-1,1)


print("After vectorizations")
print(X_train_posted_project_norm.shape, y_train.shape)
print(X_cv_posted_project_norm.shape, y_cv.shape)
print(X_test_posted_project_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(69918, 1) (69918,)
(17480, 1) (17480,)
(21850, 1) (21850,)
====================================================================================================
```

3.3 Make Data Model Ready: encoding eassay, and project_title

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
# 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
```

3.3.1 encoding essay

3.3.1.1 encoding essay : BOW

In [47]:

In [47]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(project_data['clean_essay'].values)

X_train_essay_bow = vectorizer_bow_essay.transform(X_train['clean_essay'].values)
X_cv_essay_bow = vectorizer_bow_essay.transform(X_cv['clean_essay'].values)
X_test_essay_bow = vectorizer_bow_essay.transform(X_test['clean_essay'].values)

print("After vectorizations")
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)
#print(vectorizer.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     bow_features.append(i)
```

```
After vectorizations
(69918, 16623) (69918,)
(17480, 16623) (17480,)
(21850, 16623) (21850,)
====================================================================================================
```

3.3.1.2 encoding essay : TFIDF

In [48]:

```
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(project_data['clean_essay'].values)

X_train_essay_tfidf = vectorizer_tfidf_essay.transform(X_train['clean_essay'].values)
X_cv_essay_tfidf= vectorizer_tfidf_essay.transform(X_cv['clean_essay'].values)
X_test_essay_tfidf = vectorizer_tfidf_essay.transform(X_test['clean_essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
#print(vectorizer.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 16623) (69918,)
(17480, 16623) (17480,)
(21850, 16623) (21850,)
====================================================================================================
```

3.3.2 encoding titles

3.3.2.1 encoding titles : BOW

In [49]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_titles = CountVectorizer(min_df=10)
vectorizer_bow_titles.fit(project_data['clean_project_title'].values)

X_train_title_bow = vectorizer_bow_titles.transform(X_train['clean_project_title'].values)
X_cv_title_bow = vectorizer_bow_titles.transform(X_cv['clean_project_title'].values)
X_test_title_bow = vectorizer_bow_titles.transform(X_test['clean_project_title'].values)

print("After vectorizations")
print("After vectorizations")
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)
#print(vectorizer.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     bow_features.append(i)
```

```
After vectorizations
(69918, 3222) (69918,)
(17480, 3222) (17480,)
(21850, 3222) (21850,)
====================================================================================================
```

3.3.2.2 encoding titles : TFIDF

In [50]:

```
vectorizer_tfidf_titles = TfidfVectorizer(min_df=10)
vectorizer_tfidf_titles.fit(project_data['clean_project_title'].values)

X_train_title_tfidf = vectorizer_tfidf_titles.transform(X_train['clean_project_title'].values)
X_cv_title_tfidf= vectorizer_tfidf_titles.transform(X_cv['clean_project_title'].values)
X_test_title_tfidf = vectorizer_tfidf_titles.transform(X_test['clean_project_title'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
#print(vectorizer.get_feature_names())
print("="*100)

# for i in vectorizer.get_feature_names():
#     tfidf_features.append(i)
```

```
After vectorizations
(69918, 3222) (69918,)
(17480, 3222) (17480,)
(21850, 3222) (21850,)
====================================================================================================
```

3.4 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

3.4.1 Applying Naive Bayes on BOW, SET 1

In [51]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_ohe, X_train_subcategories_ohe, X_train_price_norm, X_train_posted_project_norm, X_train_essay_bow, X_train_title_bow)).tocsr()
X_cr = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_categories_ohe, X_cv_subcategories_ohe, X_cv_price_norm, X_cv_posted_project_norm, X_cv_essay_bow, X_cv_title_bow)).tocsr()
X_te = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe, X_test_subcategories_ohe, X_test_price_norm, X_test_posted_project_norm, X_test_essay_bow, X_test_title_bow)).tocsr()

print("Final Data matrix")
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

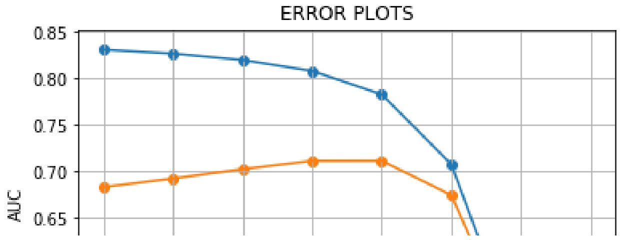
```
Final Data matrix
(69918, 19947) (69918,)
(17480, 19947) (17480,)
(21850, 19947) (21850,)
====================================================================================================
```

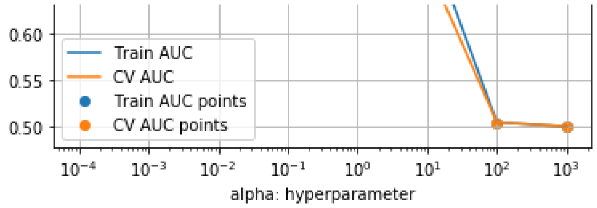
3.4.1.1 Hyper parameter tuning

In [52]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thr
esholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = [10**i for i in range(-4,4)]
for i in tqdm(a):
    nb_bow = MultinomialNB(alpha=i, fit_prior=True, class_prior = [0.5,0.5])
    nb_bow.fit(X_tr, y_train)

    y_train_pred = nb_bow.predict_log_proba(X_tr)[:,1]
    y_cv_pred = nb_bow.predict_log_proba(X_cr)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

```
100%|████████████████████████████████████████████████████████████| 8/8 [00:01<0
0:00,  5.40it/s]
```





In [53]:

```
cv_auc
```

Out[53]:

```
[0.6825824874022816,
 0.6917995285023165,
 0.7018894336942069,
 0.7107670029621393,
 0.7108594119174824,
 0.673935616793024,
 0.5043333947531061,
 0.5]
```

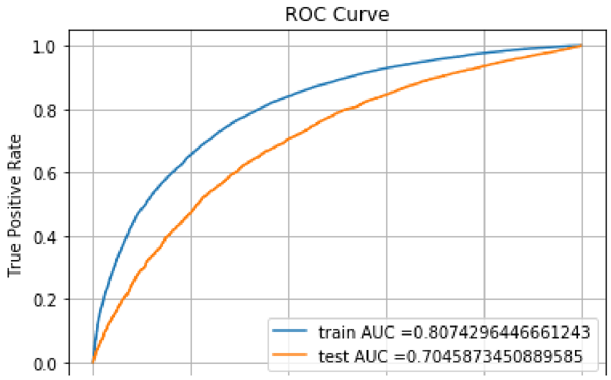
3.4.1.2 Testing the performance of the model on test data, plotting ROC Curves

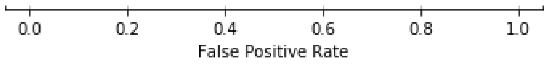
In [54]:

```
best_alpha = 0.1
```

In [55]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_
curve
from sklearn.metrics import roc_curve, auc


nb_bow = MultinomialNB(alpha=best_alpha, fit_prior=True, class_prior = [0.5,0.5])
nb_bow.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive clas
s
# not the predicted outputs

y_train_pred = nb_bow.predict_log_proba(X_tr)[:,1]
y_test_pred = nb_bow.predict_log_proba(X_te)[:,1]

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

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





In [56]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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))
    return t

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

In [57]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.5406889473211713 for threshold -0.878
```

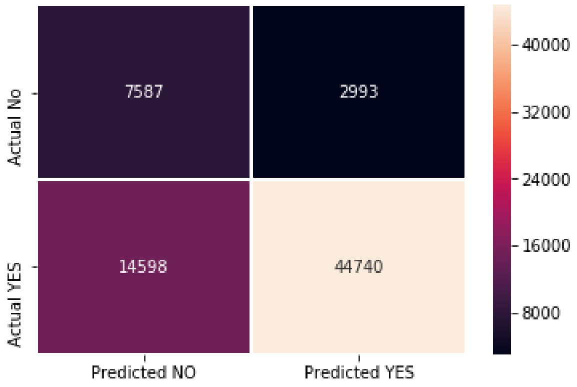
In [58]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [59]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
```

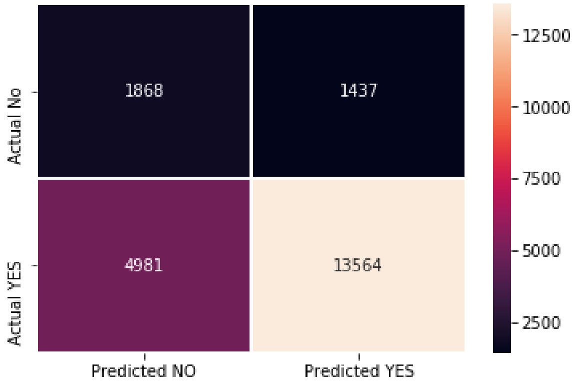
```
Train confusion matrix
```



In [60]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
Test confusion matrix
```



2.4.1.1 Top 10 important features of positive class from SET 1

In [61]:

```
bow_features = []
tfidf_features = []
```

In [62]:

```
for i in vectorizer1.get_feature_names():      # clean_school_states
    bow_features.append(i)
    tfidf_features.append(i)

for i in vectorizer2.get_feature_names():      # teacher_prefix
    bow_features.append(i)
    tfidf_features.append(i)

for i in vectorizer3.get_feature_names():      # project_grade_category
    bow_features.append(i)
    tfidf_features.append(i)

for i in vectorizer4.get_feature_names():      # project_subject_categories
    bow_features.append(i)
    tfidf_features.append(i)

for i in vectorizer5.get_feature_names():      # project_subject_subcategories
    bow_features.append(i)
    tfidf_features.append(i)

for i in vectorizer_bow_essay.get_feature_names():      # bow essay
    bow_features.append(i)

for i in vectorizer_bow_titles.get_feature_names():      # bow titles
    bow_features.append(i)



for i in vectorizer_tfidf_essay.get_feature_names():      # tfidf essay
    tfidf_features.append(i)

for i in vectorizer_tfidf_titles.get_feature_names():      # tfidf titles
    tfidf_features.append(i)
```

In [63]:

```
bow_features.append(X_train['price'])
bow_features.append(X_train['teacher_number_of_previously_posted_projects'])

tfidf_features.append(X_train['price'])
tfidf_features.append(X_train['teacher_number_of_previously_posted_projects'])
```

In [64]:

```
print(len(bow_features))
```

```
19947
```

In [65]:

```
print(len(nb_bow.feature_log_prob_[1]))
```

```
19947
```

In [66]:

```
bow_positive_feature_prob = []

for i in range(19947):
    bow_positive_feature_prob.append(nb_bow.feature_log_prob_[1,i])

positive_bow_df = pd.DataFrame({"feature_probability":bow_positive_feature_prob,"feature_names":bow_fea
tures})

positive_features_bow = positive_bow_df.sort_values(by=["feature_probability"],ascending=False)

positive_features_bow.head(10)
```

Out[66]:

	feature_probability	feature_names
14406	-3.069499	studentsnannan
13081	-4.212869	schooler
9849	-4.526102	myers
8667	-4.576655	learnings
2899	-4.600228	classrooms
14987	-4.828288	theaters
15030	-4.867808	thicker
10108	-4.870939	notable
8663	-4.919773	learner
7147	-4.945951	helper

2.4.1.2 Top 10 important features of negative class from SET 1

In [67]:

```
print(len(bow_features))
```

```
19947
```

In [68]:

```
print(len(nb_bow.feature_log_prob_[0]))
```

```
19947
```

In [69]:

```
bow_negative_feature_prob = []

for i in range(19947):
    bow_negative_feature_prob.append(nb_bow.feature_log_prob_[0,i])
    bow_negative_feature_prob.append(nb_bow.feature_log_prob_[0,i])

negative_bow_df = pd.DataFrame({"feature_probability":bow_negative_feature_prob,"feature_names":bow_features})

negative_features_bow = negative_bow_df.sort_values(by=["feature_probability"],ascending=False)

negative_features_bow.head(10)
```

Out[69]:

	feature_probability	feature_names
14406	-3.077866	studentsnannan
13081	-4.167985	schooler
8667	-4.495318	learnings
9849	-4.536267	myers
2899	-4.661638	classrooms
10108	-4.832457	notable
8663	-4.861087	learner
15030	-4.868560	thicker
7147	-4.883675	helper
14987	-4.891656	theaters

3.4.2 Applying Naive Bayes on TFIDF, SET 2

In [70]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_ohe, X_train_subcategories_ohe, X_train_price_norm, X_train_posted_project_norm, X_train_essay_tfidf, X_train_title_tfidf)).tocsr()
X_cr = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_categories_ohe, X_cv_subcategories_ohe, X_cv_price_norm, X_cv_posted_project_norm, X_cv_essay_tfidf, X_cv_title_tfidf)).tocsr()
X_te = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe, X_test_subcategories_ohe, X_test_price_norm, X_test_posted_project_norm, X_test_essay_tfidf, X_test_title_tfidf)).tocsr()

print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(69918, 19947) (69918,)
(17480, 19947) (17480,)
(21850, 19947) (21850,)
====================================================================================================
```

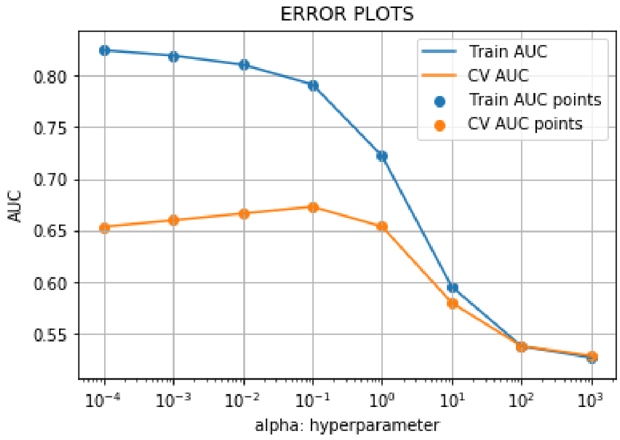
3.4.2.1 Hyper parameter tuning

In [71]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.
```

```
y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thr
esholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = [10**i for i in range(-4,4)]
for i in tqdm(a):
    nb_tfidf = MultinomialNB(alpha=i, fit_prior=True, class_prior = [0.5,0.5])
    nb_tfidf.fit(X_tr, y_train)

    y_train_pred = nb_tfidf.predict_log_proba(X_tr)[:,1]
    y_cv_pred = nb_tfidf.predict_log_proba(X_cr)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

```
100%|████████████████████████████████████████████████████████████| 8/8 [00:01<0
0:00,  5.66it/s]
```



In [72]:

```
cv_auc
```

Out[72]:

```
[0.6534147984480576,
 0.6596810117509813,
 0.6663187905580925,
 0.6728638684183819,
 0.65361576475704,
 0.5796649618680711,
 0.5376089975625313,
 0.528189885664008]
```

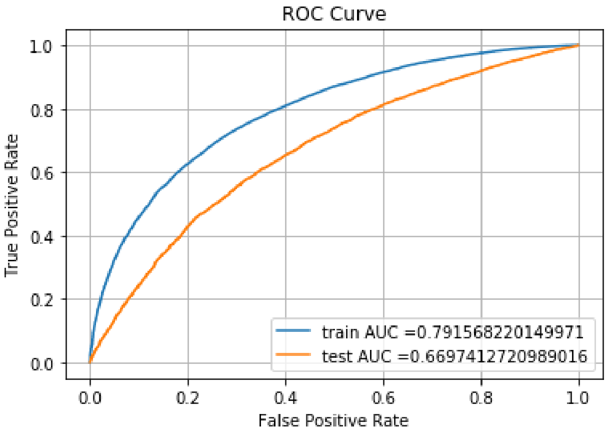
3.4.2.2 Testing the performance of the model on test data, plotting ROC Curves

In [73]:

```
best_alpha = 0.1
```

In [74]:

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


nb_tfidf = MultinomialNB(alpha=best_alpha, fit_prior=True, class_prior = [0.5,0.5])
nb_tfidf.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = nb_tfidf.predict_log_proba(X_tr)[:,1]
y_test_pred = nb_tfidf.predict_log_proba(X_te)[:,1]

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

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



In [75]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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))
    return t

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

In [76]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.5163087616799876 for threshold -0.706
```

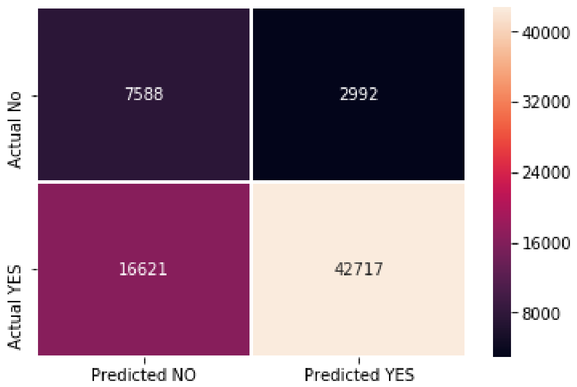
In [77]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [78]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
```

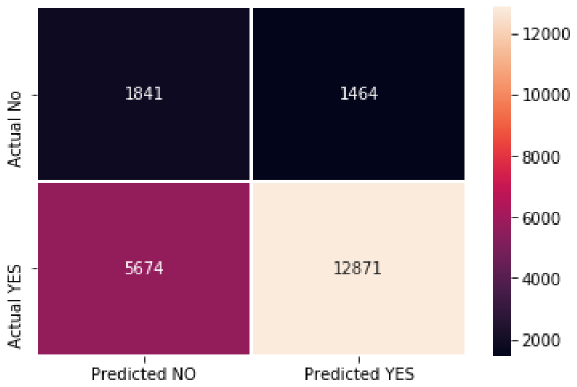
```
Train confusion matrix
```



In [79]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



2.4.2.1 Top 10 important features of positive class from SET 2

In [80]:

```
print(len(tfidf_features))
print(len(tfidf_features))
```

```
19947
```

In [81]:

```
print(len(nb_tfidf.feature_log_prob_[1]))
```

```
19947
```

In [82]:

```
tfidf_positive_feature_prob = []

for i in range(19947):
    tfidf_positive_feature_prob.append(nb_tfidf.feature_log_prob_[1,i])

positive_tfidf_df = pd.DataFrame({"feature_probability":tfidf_positive_feature_prob,"feature_names":tfi
df_features})

positive_features_tfidf = positive_tfidf_df.sort_values(by=["feature_probability"],ascending=False)

positive_features_tfidf.head(10)
```

Out[82]:

	feature_probability	feature_names
53	-3.454168	mrs
65	-3.525892	literacy_language
60	-3.718717	gradesprek_2
66	-3.789973	math_science
54	-3.844745	ms
57	-3.885740	grades3_5
87	-3.952963	literacy
89	-4.171320	mathematics
88	-4.396217	literature_writing
58	-4.678301	grades6_8

2.4.2.2 Top 10 important features of negative class from SET 2

In [83]:

```
print(len(tfidf_features))
```

```
19947
```

In [84]:

```
print(len(nb_tfidf.feature_log_prob_[0]))
```

```
19947
```

In [85]:

```
tfidf_negative_feature_prob = []

for i in range(19947):
    tfidf_negative_feature_prob.append(nb_tfidf.feature_log_prob_[0,i])

negative_tfidf_df = pd.DataFrame({"feature_probability":tfidf_negative_feature_prob,"feature_names":tfi
negative_tfidf_df = pd.DataFrame({"feature_probability":tfidf_negative_feature_prob, "feature_names":tfi
df_features})

negative_features_tfidf = negative_tfidf_df.sort_values(by=["feature_probability"],ascending=False)

negative_features_tfidf.head(10)
```

Out[85]:

	feature_probability	feature_names
53	-3.512352	mrs
65	-3.665230	literacy_language
60	-3.709953	gradesprek_2
66	-3.732257	math_science
54	-3.798758	ms
57	-3.932105	grades3_5
87	-4.156039	literacy
89	-4.158581	mathematics
88	-4.473527	literature_writing
58	-4.627255	grades6_8

3. Conclusions

In [87]:

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

from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperparameter (alpha)", "AUC"]

x.add_row(["BOW", "Auto", 0.1, 0.70458])
x.add_row(["TFIDF", "Auto", 0.1, 0.66974])


print(x)
```

```
+------------+-------+------------------------+---------+
| Vectorizer | Model | Hyperparameter (alpha) |   AUC   |
+------------+-------+------------------------+---------+
|    BOW     |  Auto |          0.1           | 0.70458 |
|   TFIDF    |  Auto |          0.1           | 0.66974 |
+------------+-------+------------------------+---------+
```