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

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

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. Preprocessing

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 & Hunge
r"]
        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('&','_') # 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 & Hunge
r"]
        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])
```

```python
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. 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 [01:30<00:
00, 1209.01it/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]:

```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 [22]:

```python
# 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, 28341.03it/s]
```

In [23]:

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

Out[23]:

```
'need move input'
```

In [24]:

```python
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
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_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
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

2.8 Adding quantity column in our dataframe

In [32]:

```
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 [33]:

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

Out[33]:

	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 [34]:

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

In [35]:

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

Out[35]:

	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

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

2.9 Preprocessing of teacher_prefix

In [36]:

```
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 [37]:

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

Out[37]:

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

2.10 Preprocessing of school_state

In [38]:

```
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 [39]:

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

Out[39]:

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

Assignment 9: RF and GBDT

Response Coding: Example

> The response tabel is built only on train dataset. For a category which is not there in train data and present in test data, we will encode them with default values Ex: in our test data if have State: D then we encode it as [0.5, 0.05]

1. **Apply both Random Forrest and GBDT on these feature sets**
    - Set 1: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)
2. **The hyper paramter tuning (Consider any two hyper parameters preferably n_estimators, max_depth)**
    - Consider the following range for hyperparameters **n_estimators** = [10, 50, 100, 150, 200, 300, 500, 1000], **max_depth** = [2, 3, 4, 5, 6, 7, 8, 9, 10]
    - Find the best hyper parameter which will give the maximum AUC value
    - find the best hyper paramter using k-fold cross validation/simple cross validation data
    - use gridsearch cv or randomsearch cv or you can write your own for loops to do this task
3. **Representation of results**
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure
      with X-axis as **n_estimators**, Y-axis as **max_depth**, and Z-axis as **AUC Score** , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

or

- 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
  seaborn heat maps with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing

AUC Score

- You can choose either of the plotting techniques: 3d plot or heat map
- 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

4. **Conclusion**

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

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link.

3. Random Forest and GBDT

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

In [60]:

```
# selecting 50k datapoints for bow and tfidf

project_data_50 = project_data.sample(n = 50000)
project_data_50.shape
```

Out[60]:

```
(50000, 14)
```

In [61]:

```
# selecting 20k datapoints for avg-w2v and tfidf-w2v

project_data_20 = project_data.sample(n = 20000)
project_data_20.shape
```

Out[61]:

```
(20000, 14)
```

In [62]:

```
# 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
from scipy.sparse import coo_matrix
```

```
X_50 = project_data_50.drop(['project_is_approved','id'], axis=1)
X_50.head(2)

y_50 = project_data_50['project_is_approved'].values


# split the data set into train and test   (for bow and tfidf)
X_train_50, X_test_50, y_train_50, y_test_50 = train_test_split(X_50, y_50, test_size=0.2,shuffle=False
)


print(X_train_50.shape, y_train_50.shape)
print(X_test_50.shape, y_test_50.shape)
```

```
(40000, 12) (40000,)
(10000, 12) (10000,)
```

In [63]:

```
X_20 = project_data_20.drop(['project_is_approved','id'], axis=1)
X_20.head(2)

y_20 = project_data_20['project_is_approved'].values


# split the data set into train and test   (for avg-w2v and tfidf-w2v)
X_train_20, X_test_20, y_train_20, y_test_20 = train_test_split(X_20, y_20, test_size=0.2,shuffle=False
)


print(X_train_20.shape, y_train_20.shape)
print(X_test_20.shape, y_test_20.shape)
```

```
(16000, 12) (16000,)
(4000, 12) (4000,)
```

3.2 Make Data Model Ready: encoding numerical, categorical features

3.2.1 encoding categorical features: School State

a) Encoding for 50k datapoints

In [65]:

```
from collections import Counter

l = X_train_50.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_50[i]==0:
        x_0.append(X_train_50.iloc[i]['clean_school_state'])

    if y_train_50[i]==1:
        x_1.append(X_train_50.iloc[i]['clean_school_state'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)
```

```
x= X_train_50.clean_school_state.value_counts()
#print(x)
```

In [66]:

```
index = X_train_50.clean_school_state.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_state = pd.DataFrame(response_table,columns=['state','class_0','class_1'])
response_df_state
```

Out[66]:

	state	class_0	class_1
0	ga	0.151617	0.848383
1	ny	0.155377	0.844623
2	wa	0.125285	0.874715
3	ar	0.157754	0.842246
4	tx	0.187547	0.812453
5	ks	0.194805	0.805195
6	pa	0.151123	0.848877
7	la	0.179458	0.820542
8	fl	0.163700	0.836300
9	ct	0.148829	0.851171
10	nc	0.147182	0.852818
11	va	0.152520	0.847480
12	sc	0.153361	0.846639
13	co	0.143229	0.856771
14	ca	0.146298	0.853702
15	nm	0.149254	0.850746
16	ok	0.161567	0.838433
17	dc	0.223350	0.776650
18	ut	0.158218	0.841782
19	mn	0.148058	0.851942
20	in	0.150685	0.849315
21	nj	0.164417	0.835583
22	mi	0.144407	0.855593
23	or	0.161702	0.838298
24	ms	0.165966	0.834034
25	oh	0.122807	0.877193
26	ri	0.131868	0.868132
27	mo	0.132400	0.867600
28	tn	0.157233	0.842767

	state	class_0	class_1
29	il	0.141338	0.858662
30	ma	0.146476	0.853524
31	nv	0.146341	0.853659
32	de	0.086331	0.913669
33	wi	0.159817	0.840183
34	id	0.190114	0.809886
35	md	0.162011	0.837989
36	al	0.150000	0.850000
37	sd	0.176471	0.823529
38	az	0.156364	0.843636
39	wv	0.116751	0.883249
40	ia	0.179039	0.820961
41	ky	0.112554	0.887446
42	hi	0.146893	0.853107
43	ne	0.200000	0.800000
44	ak	0.203252	0.796748
45	nh	0.097744	0.902256
46	mt	0.219780	0.780220
47	wy	0.184211	0.815789
48	me	0.123711	0.876289
49	vt	0.080000	0.920000
50	nd	0.137931	0.862069

In [67]:

```
train_50_state_encoded_0 = []
train_50_state_encoded_1 = []
l = X_train_50.shape[0]
for i in tqdm(range(l)):
    state = X_train_50.iloc[i]['clean_school_state']
    for j in range(response_df_state.shape[0]):
        if response_df_state.iloc[j]['state']==state:
            train_50_state_encoded_0.append(response_df_state.iloc[j]['class_0'])
            train_50_state_encoded_1.append(response_df_state.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████| 40000/40000 [09:15<0
0:00, 72.04it/s]
```

In [68]:

```
train_50_state_encoded_0 = (coo_matrix(train_50_state_encoded_0)).reshape(-1,1)
train_50_state_encoded_1 = (coo_matrix(train_50_state_encoded_1)).reshape(-1,1)
```

In [69]:

```
test_50_state_encoded_0 = []
test_50_state_encoded_1 = []
l = X_test_50.shape[0]
for i in tqdm(range(l)):
    state = X_test_50.iloc[i]['clean_school_state']
    for j in range(response_df_state.shape[0]):
        if response_df_state.iloc[j]['state']==state:
            test_50_state_encoded_0.append(response_df_state.iloc[j]['class_0'])
            test_50_state_encoded_1.append(response_df_state.iloc[j]['class_1'])
```

```
100%|████████████████████████████████████████| 10000/10000 [02:23<0
0:00, 69.44it/s]
```

In [70]:

```
test_50_state_encoded_0 = (coo_matrix(test_50_state_encoded_0)).reshape(-1,1)
test_50_state_encoded_1 = (coo_matrix(test_50_state_encoded_1)).reshape(-1,1)
```

b) Encoding for 20k datapoints

In [71]:

```
X_train_20.head(2)
```

Out[71]:

	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projects	clean_categorie
71644	FL	2017-02-28 18:10:33	6	Literacy_Languag
105653	VA	2016-10-01 20:24:15	6	Literacy_Languag

In [72]:

```
type(X_train_20)
```

Out[72]:

```
pandas.core.frame.DataFrame
```

In [73]:

```
from collections import Counter

l = X_train_20.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_20[i]==0:
        x_0.append(X_train_20.iloc[i]['clean_school_state'])

    if y_train_20[i]==1:
        x_1.append(X_train_20.iloc[i]['clean_school_state'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)





x= X_train_20.clean_school_state.value_counts()
#print(x)
```

In [74]:

```
index = X_train_20.clean_school_state.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_state = pd.DataFrame(response_table,columns=['state','class_0','class_1'])
response_df_state
```

Out[74]:

	state	class_0	class_1
0	fl	0.178610	0.821390
1	va	0.165517	0.834483
2	ca	0.144060	0.855940
3	ma	0.118132	0.881868
4	sc	0.162712	0.837288
5	ks	0.133929	0.866071
6	mn	0.132979	0.867021
7	wa	0.084986	0.915014
8	in	0.143243	0.856757
9	oh	0.108824	0.891176
10	ky	0.125604	0.874396
11	nc	0.144947	0.855053
12	ny	0.141463	0.858537
13	la	0.176301	0.823699
14	pa	0.152083	0.847917
15	md	0.136752	0.863248
16	nh	0.178571	0.821429
17	nj	0.159875	0.840125
18	ct	0.146119	0.853881
19	ga	0.170213	0.829787
20	az	0.184615	0.815385
21	tx	0.180645	0.819355
22	ok	0.167614	0.832386
23	il	0.151613	0.848387
24	ar	0.154412	0.845588
25	or	0.206061	0.793939
26	nm	0.154762	0.845238
27	mi	0.161504	0.838496
28	tn	0.146939	0.853061
29	wv	0.111111	0.888889
30	al	0.133080	0.866920
31	dc	0.216216	0.783784

	state	class_0	class_1
32	ny	0.165775	0.834225
33	id	0.186275	0.813725
34	mo	0.156010	0.843990
35	ut	0.143411	0.856589
36	wi	0.173554	0.826446
37	co	0.156627	0.843373
38	me	0.175676	0.824324
39	sd	0.225000	0.775000
40	de	0.085106	0.914894
41	ia	0.171717	0.828283
42	hi	0.115942	0.884058
43	ms	0.130653	0.869347
44	ne	0.147059	0.852941
45	mt	0.219512	0.780488
46	wy	0.181818	0.818182
47	nd	0.090909	0.909091
48	ri	0.181818	0.818182
49	ak	0.219512	0.780488
50	vt	0.545455	0.454545

In [75]:

```
train_20_state_encoded_0 = []
train_20_state_encoded_1 = []
l = X_train_20.shape[0]
for i in tqdm(range(l)):
    state = X_train_20.iloc[i]['clean_school_state']
    for j in range(response_df_state.shape[0]):
        if response_df_state.iloc[j]['state']==state:
            train_20_state_encoded_0.append(response_df_state.iloc[j]['class_0'])
            train_20_state_encoded_1.append(response_df_state.iloc[j]['class_1'])
```

```
100%|████████████████████████████████████████████████████████████████| 16000/16000 [03:49<0
0:00, 69.81it/s]
```

In [168]:

```
train_20_state_encoded_0 = (coo_matrix(train_20_state_encoded_0)).reshape(-1,1)
train_20_state_encoded_1 = (coo_matrix(train_20_state_encoded_1)).reshape(-1,1)
```

In [76]:

```
test_20_state_encoded_0 = []
test_20_state_encoded_1 = []
l = X_test_20.shape[0]
for i in tqdm(range(l)):
    state = X_test_20.iloc[i]['clean_school_state']
    for j in range(response_df_state.shape[0]):
        if response_df_state.iloc[j]['state']==state:
            test_20_state_encoded_0.append(response_df_state.iloc[j]['class_0'])
            test_20_state_encoded_1.append(response_df_state.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████████████████████| 4000/4000 [00:56<0
0:00, 70.62it/s]
```

In [169]:

```
test_20_state_encoded_0 = (coo_matrix(test_20_state_encoded_0)).reshape(-1,1)
test_20_state_encoded_1 = (coo_matrix(test_20_state_encoded_1)).reshape(-1,1)
```

3.2.2 encoding categorical features: teacher prefix

a) Encoding for 50k datapoints

In [77]:

```
from collections import Counter

l = X_train_50.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_50[i]==0:
        x_0.append( X_train_50.iloc[i]['clean_teacher_prefix'])

    if y_train_50[i]==1:
        x_1.append( X_train_50.iloc[i]['clean_teacher_prefix'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)





x= X_train_50.clean_teacher_prefix.value_counts()
#print(x)
```

In [78]:

```
index = X_train_50.clean_teacher_prefix.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_prefix = pd.DataFrame(response_table,columns=['teacher_prefix','class_0','class_1'])
response_df_prefix
```

Out[78]:

	teacher_prefix	class_0	class_1
0	mrs	0.146111	0.853889
1	ms	0.158563	0.841437
2	mr	0.159237	0.840763
3	teacher	0.229746	0.770254
4	dr	0.750000	0.250000
5	nan	0.000000	1.000000

In [79]:

```
train_50_teacher_prefix_encoded_0 = []
train_50_teacher_prefix_encoded_1 = []
l = X_train_50.shape[0]
for i in tqdm(range(l)):
    prefix = X_train_50.iloc[i]['clean_teacher_prefix']
    for j in range(response_df_prefix.shape[0]):
        if response_df_prefix.iloc[j]['teacher_prefix']==prefix:
            train_50_teacher_prefix_encoded_0.append(response_df_prefix.iloc[j]['class_0'])
            train_50_teacher_prefix_encoded_1.append(response_df_prefix.iloc[j]['class_1'])
```

```
100%|███████████████████████████████████████████████████████████████████████████| 40000/40000 [01:37<00
:00, 409.21it/s]
```

In [80]:

```
train_50_teacher_prefix_encoded_0 = (coo_matrix(train_50_teacher_prefix_encoded_0)).reshape(-1,1)
train_50_teacher_prefix_encoded_1 = (coo_matrix(train_50_teacher_prefix_encoded_1)).reshape(-1,1)
```

In [81]:

```
test_50_teacher_prefix_encoded_0 = []
test_50_teacher_prefix_encoded_1 = []
l = X_test_50.shape[0]
for i in tqdm(range(l)):
    prefix = X_test_50.iloc[i]['clean_teacher_prefix']
    for j in range(response_df_prefix.shape[0]):
        if response_df_prefix.iloc[j]['teacher_prefix']==prefix:
            test_50_teacher_prefix_encoded_0.append(response_df_prefix.iloc[j]['class_0'])
            test_50_teacher_prefix_encoded_1.append(response_df_prefix.iloc[j]['class_1'])
```

```
100%|███████████████████████████████████████████████████████████████████████████| 10000/10000 [00:24<00
:00, 414.68it/s]
```

In [82]:

```
test_50_teacher_prefix_encoded_0 = (coo_matrix(test_50_teacher_prefix_encoded_0)).reshape(-1,1)
test_50_teacher_prefix_encoded_1 = (coo_matrix(test_50_teacher_prefix_encoded_1)).reshape(-1,1)
```

b) Encoding for 20k datapoints

In [83]:

```
from collections import Counter

l = X_train_20.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_20[i]==0:
        x_0.append( X_train_20.iloc[i]['clean_teacher_prefix'])

    if y_train_20[i]==1:
        x_1.append( X_train_20.iloc[i]['clean_teacher_prefix'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)
```

```
x= X_train_20.clean_teacher_prefix.value_counts()
#print(x)
```

In [84]:

```
index = X_train_20.clean_teacher_prefix.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_prefix = pd.DataFrame(response_table,columns=['teacher_prefix','class_0','class_1'])
response_df_prefix
```

Out[84]:

	teacher_prefix	class_0	class_1
0	mrs	0.150065	0.849935
1	ms	0.151892	0.848108
2	teacher	0.220896	0.779104
3	mr	0.166667	0.833333
4	dr	1.000000	0.000000

In [85]:

```
train_20_teacher_prefix_encoded_0 = []
train_20_teacher_prefix_encoded_1 = []
l = X_train_20.shape[0]
for i in tqdm(range(l)):
    prefix = X_train_20.iloc[i]['clean_teacher_prefix']
    for j in range(response_df_prefix.shape[0]):
        if response_df_prefix.iloc[j]['teacher_prefix']==prefix:
            train_20_teacher_prefix_encoded_0.append(response_df_prefix.iloc[j]['class_0'])
            train_20_teacher_prefix_encoded_1.append(response_df_prefix.iloc[j]['class_1'])
```

```
100%|████████████████████████████████████████████████████████████| 16000/16000 [00:36<00
:00, 443.91it/s]
```

In [170]:

```
train_20_teacher_prefix_encoded_0 = (coo_matrix(train_20_teacher_prefix_encoded_0)).reshape(-1,1)
train_20_teacher_prefix_encoded_1 = (coo_matrix(train_20_teacher_prefix_encoded_1)).reshape(-1,1)
```

In [86]:

```
test_20_teacher_prefix_encoded_0 = []
test_20_teacher_prefix_encoded_1 = []
l = X_test_20.shape[0]
for i in tqdm(range(l)):
    prefix = X_test_20.iloc[i]['clean_teacher_prefix']
    for j in range(response_df_prefix.shape[0]):
        if response_df_prefix.iloc[j]['teacher_prefix']==prefix:
            test_20_teacher_prefix_encoded_0.append(response_df_prefix.iloc[j]['class_0'])
            test_20_teacher_prefix_encoded_1.append(response_df_prefix.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████████████████| 4000/4000 [00:08<00
:00, 464.31it/s]
```

```
:00, 464.31it/s]
```

In [171]:

```
test_20_teacher_prefix_encoded_0 = (coo_matrix(test_20_teacher_prefix_encoded_0)).reshape(-1,1)
test_20_teacher_prefix_encoded_1 = (coo_matrix(test_20_teacher_prefix_encoded_1)).reshape(-1,1)
```

3.2.3 encoding categorical features: project_grade_category

a) Encoding for 50k datapoints

In [87]:

```
from collections import Counter

l = X_train_50.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_50[i]==0:
        x_0.append( X_train_50.iloc[i]['clean_grade_category'])

    if y_train_50[i]==1:
        x_1.append( X_train_50.iloc[i]['clean_grade_category'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)






x= X_train_50.clean_grade_category.value_counts()
#print(x)
```

In [88]:

```
index = X_train_50.clean_grade_category.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_grade = pd.DataFrame(response_table,columns=['grade','class_0','class_1'])
response_df_grade
```

Out[88]:

	grade	class_0	class_1
0	Grades3_5	0.145935	0.854065
1	GradesPreK_2	0.155079	0.844921
2	Grades6_8	0.158692	0.841308
3	Grades9_12	0.165629	0.834371

In [89]:

```
train_50_grade_encoded_0 = []
train_50_grade_encoded_1 = []
l = X_train_50.shape[0]
for i in tqdm(range(l)):
    grade = X_train_50.iloc[i]['clean_grade_category']
    for j in range(response_df_grade.shape[0]):
        if response_df_grade.iloc[j]['grade']==grade:
            train_50_grade_encoded_0.append(response_df_grade.iloc[j]['class_0'])
            train_50_grade_encoded_1.append(response_df_grade.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 40000/40000 [01:33<00
:00, 427.10it/s]
```

In [90]:

```
train_50_grade_encoded_0 = (coo_matrix(train_50_grade_encoded_0)).reshape(-1,1)
train_50_grade_encoded_1 = (coo_matrix(train_50_grade_encoded_1)).reshape(-1,1)
```

In [91]:

```
test_50_grade_encoded_0 = []
test_50_grade_encoded_1 = []
l = X_test_50.shape[0]
for i in tqdm(range(l)):
    grade = X_test_50.iloc[i]['clean_grade_category']
    for j in range(response_df_grade.shape[0]):
        if response_df_grade.iloc[j]['grade']==grade:
            test_50_grade_encoded_0.append(response_df_grade.iloc[j]['class_0'])
            test_50_grade_encoded_1.append(response_df_grade.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 10000/10000 [00:21<00
:00, 456.26it/s]
```

In [92]:

```
test_50_grade_encoded_0 = (coo_matrix(test_50_grade_encoded_0)).reshape(-1,1)
test_50_grade_encoded_1 = (coo_matrix(test_50_grade_encoded_1)).reshape(-1,1)
```

b) Encoding for 20k datapoints

In [93]:

```
from collections import Counter

l = X_train_20.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_20[i]==0:
        x_0.append( X_train_20.iloc[i]['clean_grade_category'])

    if y_train_20[i]==1:
        x_1.append( X_train_20.iloc[i]['clean_grade_category'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)
```

```
x= X_train_20.clean_grade_category.value_counts()
#print(x)
```

In [94]:

```
index = X_train_20.clean_grade_category.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_grade = pd.DataFrame(response_table,columns=['grade','class_0','class_1'])
response_df_grade
```

Out[94]:

	grade	class_0	class_1
0	Grades3_5	0.150799	0.849201
1	GradesPreK_2	0.152194	0.847806
2	Grades9_12	0.154755	0.845245
3	Grades6_8	0.164803	0.835197

In [95]:

```
train_20_grade_encoded_0 = []
train_20_grade_encoded_1 = []
l = X_train_20.shape[0]
for i in tqdm(range(l)):
    grade = X_train_20.iloc[i]['clean_grade_category']
    for j in range(response_df_grade.shape[0]):
        if response_df_grade.iloc[j]['grade']==grade:
            train_20_grade_encoded_0.append(response_df_grade.iloc[j]['class_0'])
            train_20_grade_encoded_1.append(response_df_grade.iloc[j]['class_1'])
```

```
100%|████████████████████████████████████████████████████████████| 16000/16000 [00:34<00
:00, 470.42it/s]
```

In [172]:

```
train_20_grade_encoded_0 = (coo_matrix(train_20_grade_encoded_0)).reshape(-1,1)
train_20_grade_encoded_1 = (coo_matrix(train_20_grade_encoded_1)).reshape(-1,1)
```

In [96]:

```
test_20_grade_encoded_0 = []
test_20_grade_encoded_1 = []
l = X_test_20.shape[0]
for i in tqdm(range(l)):
    grade = X_test_20.iloc[i]['clean_grade_category']
    for j in range(response_df_grade.shape[0]):
        if response_df_grade.iloc[j]['grade']==grade:
            test_20_grade_encoded_0.append(response_df_grade.iloc[j]['class_0'])
            test_20_grade_encoded_1.append(response_df_grade.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████████████████| 4000/4000 [00:08<00
:00, 489.96it/s]
```

In [173]:

```
test_20_grade_encoded_0 = (coo_matrix(test_20_grade_encoded_0)).reshape(-1,1)
test_20_grade_encoded_1 = (coo_matrix(test_20_grade_encoded_1)).reshape(-1,1)
```

3.2.4 encoding categorical features: project_subject_categories

a) Encoding for 50k datapoints

In [97]:

```
from collections import Counter

l = X_train_50.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_50[i]==0:
        x_0.append( X_train_50.iloc[i]['clean_categories'])

    if y_train_50[i]==1:
        x_1.append( X_train_50.iloc[i]['clean_categories'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)






x= X_train_50.clean_categories.value_counts()
#print(x)
```

In [98]:

```
index = X_train_50.clean_categories.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_categories = pd.DataFrame(response_table,columns=['categories','class_0','class_1'])
response_df_categories
```

Out[98]:

	categories	class_0	class_1
0	Literacy_Language	0.133118	0.866882
1	Math_Science AppliedLearning	0.177528	0.822472
2	Literacy_Language Math_Science	0.131951	0.868049
3	Literacy_Language SpecialNeeds	0.146275	0.853725
4	Math_Science	0.178399	0.821601
5	Math_Science SpecialNeeds	0.172205	0.827795

	categories	class_0	class_1
6	Math_Science Music_Arts	0.159869	0.840131
7	Music_Arts	0.154292	0.845708
8	Math_Science Literacy_Language	0.169533	0.830467
9	AppliedLearning Literacy_Language	0.148379	0.851621
10	Health_Sports	0.162547	0.837453
11	AppliedLearning Math_Science	0.173127	0.826873
12	SpecialNeeds	0.183673	0.816327
13	Health_Sports Literacy_Language	0.154982	0.845018
14	History_Civics Literacy_Language	0.108911	0.891089
15	Health_Sports AppliedLearning	0.144928	0.855072
16	AppliedLearning	0.178672	0.821328
17	Health_Sports SpecialNeeds	0.149701	0.850299
18	AppliedLearning SpecialNeeds	0.203738	0.796262
19	AppliedLearning Health_Sports	0.173160	0.826840
20	History_Civics Music_Arts	0.218487	0.781513
21	Warmth Care_Hunger	0.065476	0.934524
22	AppliedLearning Music_Arts	0.210526	0.789474
23	History_Civics	0.167414	0.832586
24	Literacy_Language Music_Arts	0.161527	0.838473
25	Literacy_Language AppliedLearning	0.134529	0.865471
26	Math_Science History_Civics	0.142222	0.857778
27	History_Civics Math_Science	0.115044	0.884956
28	Literacy_Language History_Civics	0.108974	0.891026
29	Literacy_Language Health_Sports	0.129032	0.870968
30	Health_Sports Music_Arts	0.100000	0.900000
31	Music_Arts SpecialNeeds	0.169811	0.830189
32	History_Civics SpecialNeeds	0.125000	0.875000
33	AppliedLearning History_Civics	0.189189	0.810811
34	Health_Sports Math_Science	0.178218	0.821782
35	Health_Sports History_Civics	0.000000	1.000000
36	SpecialNeeds Health_Sports	0.187500	0.812500
37	History_Civics AppliedLearning	0.222222	0.777778
38	SpecialNeeds Music_Arts	0.183333	0.816667
39	Math_Science Health_Sports	0.202797	0.797203
40	Music_Arts Warmth Care_Hunger	1.000000	0.000000
41	Music_Arts History_Civics	0.142857	0.857143
42	Music_Arts Health_Sports	0.166667	0.833333
43	Literacy_Language Warmth Care_Hunger	0.000000	1.000000
44	SpecialNeeds Warmth Care_Hunger	0.500000	0.500000
45	AppliedLearning Warmth Care_Hunger	0.200000	0.800000
46	History_Civics Health_Sports	0.000000	1.000000
47	Math_Science Warmth Care_Hunger	0.500000	0.500000
48	Music_Arts AppliedLearning	0.000000	1.000000
49	Health_Sports Warmth Care_Hunger	0.000000	1.000000

In [99]:

```
train_50_categories_encoded_0 = []
train_50_categories_encoded_1 = []
l = X_train_50.shape[0]
for i in tqdm(range(l)):
    categories = X_train_50.iloc[i]['clean_categories']
    for j in range(response_df_categories.shape[0]):
        if response_df_categories.iloc[j]['categories']==categories:
            train_50_categories_encoded_0.append(response_df_categories.iloc[j]['class_0'])
            train_50_categories_encoded_1.append(response_df_categories.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 40000/40000 [10:31<0
0:00, 63.30it/s]
```

In [100]:

```
train_50_categories_encoded_0 = (coo_matrix(train_50_categories_encoded_0)).reshape(-1,1)
train_50_categories_encoded_1 = (coo_matrix(train_50_categories_encoded_1)).reshape(-1,1)
```

In [101]:

```
test_50_categories_encoded_0 = []
test_50_categories_encoded_1 = []
l = X_test_50.shape[0]
for i in tqdm(range(l)):
    categories = X_test_50.iloc[i]['clean_categories']
    for j in range(response_df_categories.shape[0]):
        if response_df_categories.iloc[j]['categories']==categories:
            test_50_categories_encoded_0.append(response_df_categories.iloc[j]['class_0'])
            test_50_categories_encoded_1.append(response_df_categories.iloc[j]['class_1'])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 10000/10000 [02:33<0
0:00, 65.02it/s]
```

In [102]:

```
test_50_categories_encoded_0 = (coo_matrix(test_50_categories_encoded_0)).reshape(-1,1)
test_50_categories_encoded_1 = (coo_matrix(test_50_categories_encoded_1)).reshape(-1,1)
```

b) Encoding for 20k datapoints

In [193]:

```
from collections import Counter

l = X_train_20.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_20[i]==0:
        x_0.append( X_train_20.iloc[i]['clean_categories'])

    if y_train_20[i]==1:
        x_1.append( X_train_20.iloc[i]['clean_categories'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)
```

```
x= X_train_20.clean_categories.value_counts()
#print(x)
```

In [194]:

```
index = X_train_20.clean_categories.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_categories = pd.DataFrame(response_table,columns=['categories','class_0','class_1'])
response_df_categories
```

Out[194]:

	categories	class_0	class_1
0	Literacy_Language	0.126355	0.873645
1	Math_Science	0.193160	0.806840
2	Literacy_Language SpecialNeeds	0.172117	0.827883
3	AppliedLearning	0.180243	0.819757
4	Math_Science Health_Sports	0.243243	0.756757
5	Literacy_Language Math_Science	0.133524	0.866476
6	Health_Sports SpecialNeeds	0.099502	0.900498
7	SpecialNeeds	0.171157	0.828843
8	Math_Science History_Civics	0.154639	0.845361
9	Music_Arts	0.156863	0.843137
10	AppliedLearning Literacy_Language	0.102310	0.897690
11	Math_Science Literacy_Language	0.146628	0.853372
12	Health_Sports Literacy_Language	0.216981	0.783019
13	Math_Science Music_Arts	0.157258	0.842742
14	AppliedLearning Health_Sports	0.235955	0.764045
15	Literacy_Language AppliedLearning	0.147059	0.852941
16	Health_Sports	0.158954	0.841046
17	History_Civics Literacy_Language	0.087179	0.912821
18	Warmth Care_Hunger	0.073684	0.926316
19	Literacy_Language History_Civics	0.086957	0.913043
20	Literacy_Language Music_Arts	0.168033	0.831967
21	Math_Science AppliedLearning	0.190217	0.809783
22	History_Civics	0.160714	0.839286
23	AppliedLearning Music_Arts	0.214286	0.785714
24	Math_Science SpecialNeeds	0.172932	0.827068
25	History_Civics Math_Science	0.222222	0.777778
26	History_Civics SpecialNeeds	0.150000	0.850000
27	AppliedLearning Math_Science	0.169231	0.830769
28	AppliedLearning SpecialNeeds	0.142857	0.857143

	categories	class_0	class_1
29	Health_Sports AppliedLearning	0.000000	1.000000
30	AppliedLearning History_Civics	0.120000	0.880000
31	SpecialNeeds Music_Arts	0.166667	0.833333
32	History_Civics Music_Arts	0.291667	0.708333
33	Health_Sports Music_Arts	0.071429	0.928571
34	Music_Arts SpecialNeeds	0.388889	0.611111
35	History_Civics Health_Sports	0.200000	0.800000
36	History_Civics AppliedLearning	0.400000	0.600000
37	Literacy_Language Health_Sports	0.000000	1.000000
38	Health_Sports History_Civics	0.000000	1.000000
39	Health_Sports Math_Science	0.257143	0.742857
40	Music_Arts Health_Sports	0.333333	0.666667
41	Health_Sports Warmth Care_Hunger	0.000000	1.000000
42	SpecialNeeds Health_Sports	0.000000	1.000000
43	Math_Science Warmth Care_Hunger	1.000000	0.000000
44	Music_Arts History_Civics	0.000000	1.000000
45	SpecialNeeds Warmth Care_Hunger	0.666667	0.333333
46	Music_Arts AppliedLearning	0.000000	1.000000
47	AppliedLearning Warmth Care_Hunger	0.000000	1.000000
48	Literacy_Language Warmth Care_Hunger	1.000000	0.000000

In [195]:

```
train_20_categories_encoded_0 = []
train_20_categories_encoded_1 = []
l = X_train_20.shape[0]
for i in range(l):
    categories = X_train_20.iloc[i]['clean_categories']
    for j in range(response_df_categories.shape[0]):
        if response_df_categories.iloc[j]['categories']==categories:
            train_20_categories_encoded_0.append(response_df_categories.iloc[j]['class_0'])
            train_20_categories_encoded_1.append(response_df_categories.iloc[j]['class_1'])
            break
```

In [196]:

```
print(len(train_20_categories_encoded_0))
print(len(train_20_categories_encoded_1))
```

```
16000
16000
```

In [197]:

```
train_20_categories_encoded_0 = (coo_matrix(train_20_categories_encoded_0)).reshape(-1,1)
train_20_categories_encoded_1 = (coo_matrix(train_20_categories_encoded_1)).reshape(-1,1)
```

In [198]:

```
test_20_categories_encoded_0 = []
test_20_categories_encoded_1 = []
l = X_test_20.shape[0]
for i in range(l):
    categories = X_test_20.iloc[i]['clean_categories']
    for j in range(response_df_categories.shape[0]):
        if response_df_categories.iloc[j]['categories']==categories:
            test_20_categories_encoded_0.append(response_df_categories.iloc[j]['class_0'])
            test_20_categories_encoded_1.append(response_df_categories.iloc[j]['class_1'])
            break

    else:

        test_20_categories_encoded_0.append(0.5)
        test_20_categories_encoded_1.append(0.5)
```

In [199]:

```
print(len(test_20_categories_encoded_0))
print(len(test_20_categories_encoded_1))
```

```
4000
4000
```

In [200]:

```
test_20_categories_encoded_0 = (coo_matrix(test_20_categories_encoded_0)).reshape(-1,1)
test_20_categories_encoded_1 = (coo_matrix(test_20_categories_encoded_1)).reshape(-1,1)
```

3.2.5 encoding categorical features: project_subject_subcategories

a) Encoding for 50k datapoints

In [107]:

```
from collections import Counter

l = X_train_50.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_50[i]==0:
        x_0.append( X_train_50.iloc[i]['clean_subcategories'])

    if y_train_50[i]==1:
        x_1.append( X_train_50.iloc[i]['clean_subcategories'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)





x= X_train_50.clean_subcategories.value_counts()
#print(x)
```

In [108]:

```
index = X_train_50.clean_subcategories.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])
```

```
#print(response_table)


response_df_subcategories = pd.DataFrame(response_table,columns=['subcategories','class_0','class_1'])
response_df_subcategories
```

Out[108]:

	subcategories	class_0	class_1
0	ESL Literature_Writing	0.175000	0.825000
1	Literature_Writing	0.136898	0.863102
2	AppliedSciences Extracurricular	0.153846	0.846154
3	Literacy Mathematics	0.130565	0.869435
4	Literacy SpecialNeeds	0.137405	0.862595
...	...	...	...
365	EarlyDevelopment History_Geography	0.000000	1.000000
366	EarlyDevelopment ForeignLanguages	0.000000	1.000000
367	AppliedSciences FinancialLiteracy	0.000000	1.000000
368	Literature_Writing NutritionEducation	0.000000	1.000000
369	History_Geography TeamSports	0.000000	1.000000

370 rows × 3 columns

In [111]:

```
train_50_subcategories_encoded_0 = []
train_50_subcategories_encoded_1 = []
l = X_train_50.shape[0]
for i in range(l):
    subcategories = X_train_50.iloc[i]['clean_subcategories']
    for j in range(response_df_subcategories.shape[0]):
        if response_df_subcategories.iloc[j]['subcategories']==subcategories:
            train_50_subcategories_encoded_0.append(response_df_subcategories.iloc[j]['class_0'])
            train_50_subcategories_encoded_1.append(response_df_subcategories.iloc[j]['class_1'])
            break
```

In [112]:

```
print(len(train_50_subcategories_encoded_0))
print(len(train_50_subcategories_encoded_1))
```

```
40000
40000
```

In [113]:

```
train_50_subcategories_encoded_0 = (coo_matrix(train_50_subcategories_encoded_0)).reshape(-1,1)
train_50_subcategories_encoded_1 = (coo_matrix(train_50_subcategories_encoded_1)).reshape(-1,1)
```

In [114]:

```
test_50_subcategories_encoded_0 = []
test_50_subcategories_encoded_1 = []
l = X_test_50.shape[0]
for i in range(l):
    subcategories = X_test_50.iloc[i]['clean_subcategories']
    for j in range(response_df_subcategories.shape[0]):
        if response_df_subcategories.iloc[j]['subcategories']==subcategories:
            test_50_subcategories_encoded_0.append(response_df_subcategories.iloc[j]['class_0'])
            test_50_subcategories_encoded_1.append(response_df_subcategories.iloc[j]['class_1'])
            break

    else:
        test_50_subcategories_encoded_0.append(0.5)
        test_50_subcategories_encoded_1.append(0.5)
```

In [115]:

```
print(len(test_50_subcategories_encoded_0))
print(len(test_50_subcategories_encoded_1))
```

```
10000
10000
```

In [116]:

```
test_50_subcategories_encoded_0 = (coo_matrix(test_50_subcategories_encoded_0)).reshape(-1,1)
test_50_subcategories_encoded_1 = (coo_matrix(test_50_subcategories_encoded_1)).reshape(-1,1)
```

b) Encoding for 20k datapoints

In [117]:

```
from collections import Counter

l = X_train_20.shape[0]

x_0 = []
x_1 = []


for i in range(l):
    if y_train_20[i]==0:
        x_0.append( X_train_20.iloc[i]['clean_subcategories'])

    if y_train_20[i]==1:
        x_1.append( X_train_20.iloc[i]['clean_subcategories'])


x_0 = Counter(x_0)
x_1 = Counter(x_1)

#print(x_0)
#print(x_1)

x= X_train_20.clean_subcategories.value_counts()
#print(x)
```

In [118]:

```
index = X_train_20.clean_subcategories.unique()
#print(index)

response_table = []

for i in index:
    response_table.append([i,(x_0[i]/x[i]),(x_1[i]/x[i])])

#print(response_table)


response_df_subcategories = pd.DataFrame(response_table,columns=['subcategories','class_0','class_1'])
response_df_subcategories
```

Out[118]:

	subcategories	class_0	class_1

	subcategories	class_0	class_1
0	Literacy Literature_Writing	0.126437	0.873563
1	EnvironmentalScience Health_LifeScience	0.124138	0.875862
2	AppliedSciences	0.208651	0.791349
3	Literacy SpecialNeeds	0.133333	0.866667
4	Other	0.150794	0.849206
...	...	...	...
319	AppliedSciences FinancialLiteracy	0.000000	1.000000
320	Civics_Government VisualArts	0.000000	1.000000
321	EnvironmentalScience Gym_Fitness	1.000000	0.000000
322	Extracurricular SocialSciences	0.000000	1.000000
323	Literacy Warmth Care_Hunger	1.000000	0.000000

324 rows × 3 columns

In [119]:

```
train_20_subcategories_encoded_0 = []
train_20_subcategories_encoded_1 = []
l = X_train_20.shape[0]
for i in range(l):
    subcategories = X_train_20.iloc[i]['clean_subcategories']
    for j in range(response_df_subcategories.shape[0]):
        if response_df_subcategories.iloc[j]['subcategories']==subcategories:
            train_20_subcategories_encoded_0.append(response_df_subcategories.iloc[j]['class_0'])
            train_20_subcategories_encoded_1.append(response_df_subcategories.iloc[j]['class_1'])
            break
```

In [120]:

```
print(len(train_20_subcategories_encoded_0))
print(len(train_20_subcategories_encoded_1))
```

```
16000
16000
```

In [176]:

```
train_20_subcategories_encoded_0 = (coo_matrix(train_20_subcategories_encoded_0)).reshape(-1,1)
train_20_subcategories_encoded_1 = (coo_matrix(train_20_subcategories_encoded_1)).reshape(-1,1)
```

In [121]:

```
test_20_subcategories_encoded_0 = []
test_20_subcategories_encoded_1 = []
l = X_test_20.shape[0]
for i in range(l):
    subcategories = X_test_20.iloc[i]['clean_subcategories']
    for j in range(response_df_subcategories.shape[0]):
        if response_df_subcategories.iloc[j]['subcategories']==subcategories:
            test_20_subcategories_encoded_0.append(response_df_subcategories.iloc[j]['class_0'])
            test_20_subcategories_encoded_1.append(response_df_subcategories.iloc[j]['class_1'])
            break

    else:
        test_20_subcategories_encoded_0.append(0.5)
        test_20_subcategories_encoded_1.append(0.5)
```

In [122]:

```
print(len(test_20_subcategories_encoded_0))
print(len(test_20_subcategories_encoded_1))
```

```
4000
4000
```

In [123]:

```
X_test_20.shape[0]
```

Out[123]:

```
4000
```

In [177]:

```
test_20_subcategories_encoded_0 = (coo_matrix(test_20_subcategories_encoded_0)).reshape(-1,1)
test_20_subcategories_encoded_1 = (coo_matrix(test_20_subcategories_encoded_1)).reshape(-1,1)
```

3.2.6 encoding numerical feature: price

a) Encoding for 50k datapoints

In [124]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# 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.
scaler.fit(X_train_50['price'].values.reshape(-1,1))

X_train_50_price_scaler = scaler.transform(X_train_50['price'].values.reshape(-1,1))
#X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(1,-1))
X_test_50_price_scaler = scaler.transform(X_test_50['price'].values.reshape(-1,1))


# X_train_price_scaler = X_train_price_scaler.reshape(-1,1)
# #X_cv_price_norm = X_cv_price_norm.reshape(-1,1)
# X_test_price_scaler = X_test_price_scaler.reshape(-1,1)


print("After vectorizations")
print(X_train_50_price_scaler.shape, y_train_50.shape)
#print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_50_price_scaler.shape, y_test_50.shape)
print("="*100)
```

```
After vectorizations
(40000, 1) (40000,)
(10000, 1) (10000,)
====================================================================================================
```

b) Encoding for 20k datapoints

In [125]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# 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.
scaler.fit(X_train_20['price'].values.reshape(-1,1))
```

```
X_train_20_price_scaler = scaler.transform(X_train_20['price'].values.reshape(-1,1))
#X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(1,-1))
X_test_20_price_scaler = scaler.transform(X_test_20['price'].values.reshape(-1,1))


# X_train_price_scaler = X_train_price_scaler.reshape(-1,1)
# #X_cv_price_norm = X_cv_price_norm.reshape(-1,1)
# X_test_price_scaler = X_test_price_scaler.reshape(-1,1)


print("After vectorizations")
print(X_train_20_price_scaler.shape, y_train_20.shape)
#print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_20_price_scaler.shape, y_test_20.shape)
print("="*100)
```

```
After vectorizations
(16000, 1) (16000,)
(4000, 1) (4000,)
====================================================================================================
```

3.2.7 encoding numerical feature: teacher_number_of_previously_posted_projects

a) Encoding for 50k datapoints

In [126]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# 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.
scaler.fit(X_train_50['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_50_posted_project_scaler = scaler.transform(X_train_50['teacher_number_of_previously_posted_pro
jects'].values.reshape(-1,1))
#X_cv_posted_project_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].v
alues.reshape(1,-1))
X_test_50_posted_project_scaler = scaler.transform(X_test_50['teacher_number_of_previously_posted_proje
cts'].values.reshape(-1,1))

# X_train_posted_project_scaler = X_train_posted_project_scaler.reshape(-1,1)
# #X_cv_posted_project_norm = X_cv_posted_project_norm.reshape(-1,1)
# X_test_posted_project_scaler = X_test_posted_project_scaler.reshape(-1,1)


print("After vectorizations")
print(X_train_50_posted_project_scaler.shape, y_train_50.shape)
#print(X_cv_posted_project_norm.shape, y_cv.shape)
print(X_test_50_posted_project_scaler.shape, y_test_50.shape)
print("="*100)
```

```
After vectorizations
(40000, 1) (40000,)
(10000, 1) (10000,)
====================================================================================================
```

b) Encoding for 20k datapoints

In [127]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# 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.
scaler.fit(X_train_20['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_20_posted_project_scaler = scaler.transform(X_train_20['teacher_number_of_previously_posted_pro
jects'].values.reshape(-1,1))
#X_cv_posted_project_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].v
alues.reshape(1,-1))
X_test_20_posted_project_scaler = scaler.transform(X_test_20['teacher_number_of_previously_posted_proje
cts'].values.reshape(-1,1))

# X_train_posted_project_scaler = X_train_posted_project_scaler.reshape(-1,1)
# #X_cv_posted_project_norm = X_cv_posted_project_norm.reshape(-1,1)
# X_test_posted_project_scaler = X_test_posted_project_scaler.reshape(-1,1)


print("After vectorizations")
print(X_train_20_posted_project_scaler.shape, y_train_20.shape)
#print(X_cv_posted_project_norm.shape, y_cv.shape)
print(X_test_20_posted_project_scaler.shape, y_test_20.shape)
print("="*100)
```

```
After vectorizations
(16000, 1) (16000,)
(4000, 1) (4000,)
====================================================================================================
```

3.3 Make Data Model Ready: encoding eassay, and project_title

In [ ]:

```
# 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 [128]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(project_data_50['clean_essay'].values)

X_train_essay_bow = vectorizer.transform(X_train_50['clean_essay'].values)
#X_cv_essay_bow = vectorizer.transform(X_cv['clean_essay'].values)
X_test_essay_bow = vectorizer.transform(X_test_50['clean_essay'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train_50.shape)
#print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test_50.shape)
#print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 12224) (40000,)
(10000, 12224) (10000,)
====================================================================================================
```

3.3.1.2 encoding essay : TFIDF

In [129]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(project_data_50['clean_essay'].values)

X_train_essay_tfidf = vectorizer.transform(X_train_50['clean_essay'].values)
#X_cv_essay_tfidf= vectorizer.transform(X_cv['clean_essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test_50['clean_essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train_50.shape)
#print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test_50.shape)
#print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 12224) (40000,)
(10000, 12224) (10000,)
====================================================================================================
```

3.3.1.3 encoding essay : AVG W2V

In [294]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-an
d-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 [295]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essay_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in (X_train_20['clean_essay'].values): # 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_essay_train.append(vector)

print(len(avg_w2v_essay_train))
print(len(avg_w2v_essay_train[0]))
#print(avg_w2v_essay_train[0])
```

```
16000
300
```

In [296]:

```
avg_w2v_essay_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in (X_test_20['clean_essay'].values): # 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
        vector /= cnt_words
    avg_w2v_essay_test.append(vector)
```

3.3.1.4 encoding essay : TFIDF W2V

In [133]:

```
# Similarly you can vectorize for essay

tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train_20['clean_essay'].values)
# 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 [292]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
essay_tfidf_w2v_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in (X_train_20['clean_essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_train.append(vector)

print(len(essay_tfidf_w2v_train))
print(len(essay_tfidf_w2v_train[0]))
```

```
16000
300
```

In [293]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
essay_tfidf_w2v_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in (X_test_20['clean_essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_test.append(vector)

print(len(essay_tfidf_w2v_test))
print(len(essay_tfidf_w2v_test[0]))
```

```
4000
300
```

3.3.2 encoding titles

3.3.2.1 encoding titles : BOW

In [136]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(project_data_50['clean_project_title'].values)

X_train_title_bow = vectorizer.transform(X_train_50['clean_project_title'].values)
#X_cv_title_bow = vectorizer.transform(X_cv['clean_project_title'].values)
X_test_title_bow = vectorizer.transform(X_test_50['clean_project_title'].values)

print("After vectorizations")
print(X_train_title_bow.shape, y_train_50.shape)
#print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test_50.shape)
#print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 2008) (40000,)
(10000, 2008) (10000,)
====================================================================================================
```

3.3.2.2 encoding titles : TFIDF

In [137]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(project_data_50['clean_project_title'].values)

X_train_title_tfidf = vectorizer.transform(X_train_50['clean_project_title'].values)
#X_cv_title_tfidf= vectorizer.transform(X_cv['clean_project_title'].values)
X_test_title_tfidf = vectorizer.transform(X_test_50['clean_project_title'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train_50.shape)
#print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test_50.shape)
#print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(40000, 2008) (40000,)
(10000, 2008) (10000,)
====================================================================================================
```

3.3.2.3 encoding titles : AVG W2V

In [138]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-an
d-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 [140]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_title_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train_20['clean_project_title'].values): # 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_title_train.append(vector)

print(len(avg_w2v_title_train))
print(len(avg_w2v_title_train[0]))
#print(avg_w2v_title_train[0])
```

```
  0%|                                                                                 | 0/16000
[00:00<?, ?it/s]

 14%|██████████                                                             | 2204/16000 [00:00<00:0
0, 21834.24it/s]

 27%|████████████████████▌                                                  | 4395/16000 [00:00<00:0
0, 21794.51it/s]

 47%|███████████████████████████████████▏                                   | 7441/16000 [00:00<00:0
0, 23776.25it/s]

 64%|████████████████████████████████████████████████▌                     | 10317/16000 [00:00<00:0
0, 25018.50it/s]

 78%|██████████████████████████████████████████████████████████▏           | 12462/16000 [00:00<00:0
0, 23754.69it/s]

100%|██████████████████████████████████████████████████████████████████████| 16000/16000 [00:00<00:0
0, 25290.89it/s]
```

```
16000
300
```

In [141]:

```
avg_w2v_title_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test_20['clean_project_title'].values): # 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_title_test.append(vector)
```

```
  0%|                                                                                  | 0/4000
[00:00<?, ?it/s]

100%|████████████████████████████████████████████████████████████████████████████| 4000/4000 [00:00<00:0
0, 21870.34it/s]
```

3.3.2.4 encoding titles : TFIDF W2V

In [142]:

```
# Similarly you can vectorize for title also

tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train_20['clean_project_title'].values)
# 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 [143]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
title_tfidf_w2v_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train_20['clean_project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_train.append(vector)

print(len(title_tfidf_w2v_train))
print(len(title_tfidf_w2v_train[0]))
```

```
  0%|                                                                          | 0/16000
[00:00<?, ?it/s]

  7%|█████                                                              | 1187/16000 [00:00<00:0
1, 11759.30it/s]

 20%|██████████████▌                                                    | 3124/16000 [00:00<00:0
0, 13304.76it/s]

 30%|██████████████████████                                             | 4880/16000 [00:00<00:0
0, 14314.61it/s]

 42%|██████████████████████████████▌                                    | 6690/16000 [00:00<00:0
0, 15236.67it/s]

 53%|██████████████████████████████████████▌                            | 8427/16000 [00:00<00:0
0, 15778.94it/s]

 62%|█████████████████████████████████████████████                      | 9858/16000 [00:00<00:0
0, 15212.67it/s]

 72%|████████████████████████████████████████████████████▌             | 11500/16000 [00:00<00:0
0, 15514.14it/s]

 82%|████████████████████████████████████████████████████████████      | 13193/16000 [00:00<00:0
0, 15871.34it/s]

100%|█████████████████████████████████████████████████████████████████████████| 16000/16000 [00:01<00:0
0, 15850.67it/s]
```

```
16000
300
```

In [144]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
title_tfidf_w2v_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test_20['clean_project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_test.append(vector)

print(len(title_tfidf_w2v_test))
print(len(title_tfidf_w2v_test[0]))
```

```
  0%|                                                                     | 0/4000
[00:00<?, ?it/s]

 19%|██████████████                                                 | 754/4000 [00:00<00:
00, 7469.54it/s]

 57%|███████████████████████████████████████████                   | 2291/4000 [00:00<00:
00, 8817.09it/s]

100%|██████████████████████████████████████████████████████████████████████████| 4000/4000 [00:00<00:0
0, 12625.54it/s]
```

```
4000
300
```

3.4 Applying Random Forest

Apply Random Forest 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 Random Forests on BOW, SET 1

In [145]:

```
# 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_bow = hstack((train_50_state_encoded_0,train_50_state_encoded_1, train_50_teacher_prefix_encoded_0
,train_50_teacher_prefix_encoded_1, train_50_grade_encoded_0,train_50_grade_encoded_1, train_50_categor
ies_encoded_0,train_50_categories_encoded_1, train_50_subcategories_encoded_0,train_50_subcategories_en
coded_1, X_train_50_price_scaler, X_train_50_posted_project_scaler, X_train_essay_bow, X_train_title_bo
w)).tocsr()

X_te_bow = hstack((test_50_state_encoded_0,test_50_state_encoded_1, test_50_teacher_prefix_encoded_0,te
st_50_teacher_prefix_encoded_1, test_50_grade_encoded_0,test_50_grade_encoded_1, test_50_categories_enc
oded_0,test_50_categories_encoded_1, test_50_subcategories_encoded_0,test_50_subcategories_encoded_1, X
_test_50_price_scaler, X_test_50_posted_project_scaler, X_test_essay_bow, X_test_title_bow)).tocsr()

y_train_bow = y_train_50

y_test_bow = y_test_50

print("Final Data matrix")
print(X_tr_bow.shape, y_train_bow.shape)

print(X_te_bow.shape, y_test_bow.shape)
print("="*100)
```

```
Final Data matrix
(40000, 14244) (40000,)
(10000, 14244) (10000,)
====================================================================================================
```

3.4.1.1 Hyperparameter Tuning

In [146]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt



tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_bow = RandomForestClassifier(class_weight='balanced', n_jobs=-1)

#Using GridSearchCV
model_bow = GridSearchCV(clf_bow, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,return_trai
n_score=True)
model_bow.fit(X_tr_bow, y_train_bow)



print(model_bow.best_estimator_)
print(model_bow.score(X_te_bow, y_test_bow))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  5.8min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 14.9min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 22.8min finished
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)
0.7055240571785646
```

In [147]:

```
train_auc= model_bow.cv_results_['mean_train_score']
cv_auc = model_bow.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

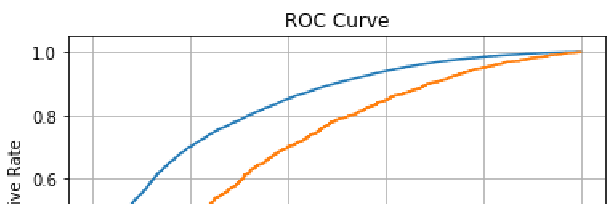
In [148]:

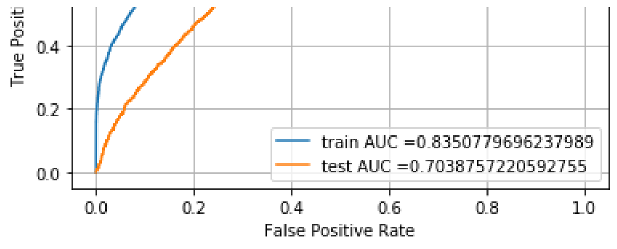
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [149]:

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

clf_bow = RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)

clf_bow.fit(X_tr_bow, y_train_bow)


y_train_pred = clf_bow.predict_proba(X_tr_bow)[:,1]
y_test_pred = clf_bow.predict_proba(X_te_bow)[:,1]





train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_bow, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_bow, 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 [150]:

```
# 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 [151]:

```
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) 0.5627549285023334 for threshold 0.501
```

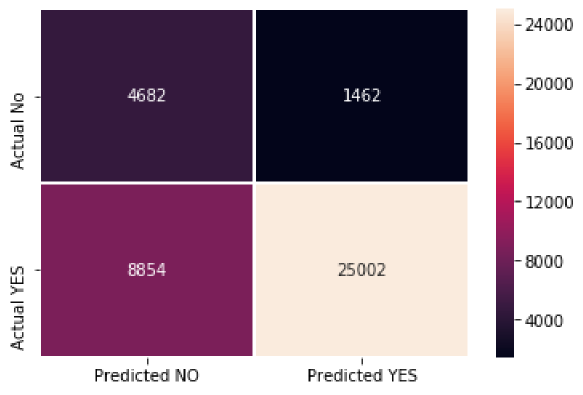
In [152]:

```
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 [153]:

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

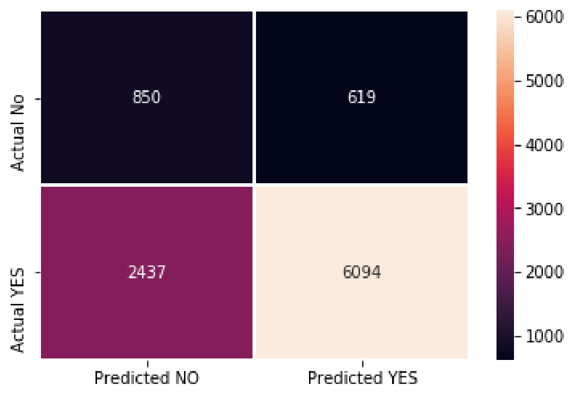
```
Train confusion matrix
```



In [154]:

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

```
Test confusion matrix
```



3.4.2 Applying Random Forests on TFIDF, SET 2

In [155]:

```
# 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_tfidf = hstack((train_50_state_encoded_0,train_50_state_encoded_1, train_50_teacher_prefix_encoded
_0,train_50_teacher_prefix_encoded_1, train_50_grade_encoded_0,train_50_grade_encoded_1, train_50_categ
ories_encoded_0,train_50_categories_encoded_1, train_50_subcategories_encoded_0,train_50_subcategories_
encoded_1, X_train_50_price_scaler, X_train_50_posted_project_scaler, X_train_essay_tfidf, X_train_titl
e_tfidf)).tocsr()

X_te_tfidf = hstack((test_50_state_encoded_0,test_50_state_encoded_1, test_50_teacher_prefix_encoded_0,
test_50_teacher_prefix_encoded_1, test_50_grade_encoded_0,test_50_grade_encoded_1, test_50_categories_e
ncoded_0,test_50_categories_encoded_1, test_50_subcategories_encoded_0,test_50_subcategories_encoded_1,
X_test_50_price_scaler, X_test_50_posted_project_scaler, X_test_essay_tfidf, X_test_title_tfidf)).tocsr
()

y_train_tfidf = y_train_50

y_test_tfidf = y_test_50

print("Final Data matrix")
print(X_tr_tfidf.shape, y_train_tfidf.shape)

print(X_te_tfidf.shape, y_test_tfidf.shape)
print("="*100)
```

```
Final Data matrix
(40000, 14244) (40000,)
(10000, 14244) (10000,)
====================================================================================================
```

3.4.2.1 Hyperparameter Tuning

In [156]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
```

```
tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_tfidf = RandomForestClassifier(class_weight='balanced', n_jobs=-1)

#Using GridSearchCV
model_tfidf = GridSearchCV(clf_tfidf, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,return_
train_score=True)
model_tfidf.fit(X_tr_tfidf, y_train_tfidf)



print(model_tfidf.best_estimator_)
print(model_tfidf.score(X_te_tfidf, y_test_tfidf))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   19.5s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  4.5min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 12.9min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 21.1min finished
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)
0.7074203168375075
```

In [157]:

```
train_auc= model_tfidf.cv_results_['mean_train_score']
cv_auc = model_tfidf.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

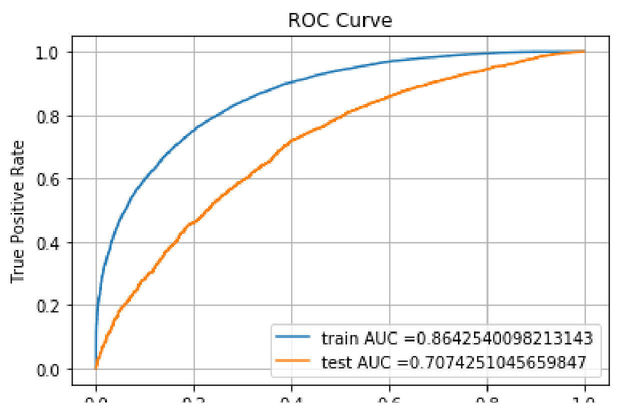
In [158]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [213]:

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

clf_tfidf = RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=10, max_features='auto',
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)

clf_tfidf.fit(X_tr_tfidf, y_train_tfidf)


y_train_pred = clf_tfidf.predict_proba(X_tr_tfidf)[:,1]
y_test_pred = clf_tfidf.predict_proba(X_te_tfidf)[:,1]




train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_tfidf, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidf, 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 [214]:

```
# 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 [215]:

```
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) 0.6038148539879071 for threshold 0.507
```

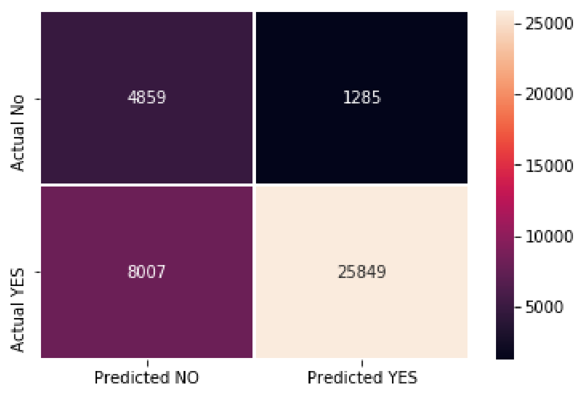
In [216]:

```
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 [217]:

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

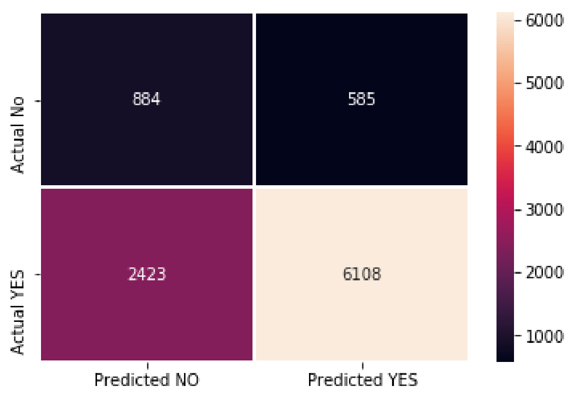
```
Train confusion matrix
```



In [218]:

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

```
Test confusion matrix
```



3.4.3 Applying Random Forests on AVG W2V, SET 3

In [202]:

```
# 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_avgw2v = hstack((train_20_state_encoded_0,train_20_state_encoded_1, train_20_teacher_prefix_encode
d_0,train_20_teacher_prefix_encoded_1, train_20_grade_encoded_0,train_20_grade_encoded_1, train_20_cate
gories_encoded_0,train_20_categories_encoded_1, train_20_subcategories_encoded_0,train_20_subcategories
_encoded_1, X_train_20_price_scaler, X_train_20_posted_project_scaler, avg_w2v_essay_train, avg_w2v_tit
le_train)).tocsr()

X_te_avgw2v = hstack((test_20_state_encoded_0,test_20_state_encoded_1, test_20_teacher_prefix_encoded_0
,test_20_teacher_prefix_encoded_1, test_20_grade_encoded_0,test_20_grade_encoded_1, test_20_categories_
encoded_0,test_20_categories_encoded_1, test_20_subcategories_encoded_0,test_20_subcategories_encoded_1
, X_test_20_price_scaler, X_test_20_posted_project_scaler, avg_w2v_essay_test, avg_w2v_title_test)).toc
sr()

y_train_avgw2v = y_train_20

y_test_avgw2v = y_test_20

print("Final Data matrix")
print(X_tr_avgw2v.shape, y_train_avgw2v.shape)

print(X_te_avgw2v.shape, y_test_avgw2v.shape)
print("="*100)
```

```
Final Data matrix
(16000, 612) (16000,)
(4000, 612) (4000,)
====================================================================================================
```

3.4.3.1 Hyperparameter Tuning

In [203]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt



tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_avgw2v = RandomForestClassifier(class_weight='balanced', n_jobs=-1)

#Using GridSearchCV
model_avgw2v = GridSearchCV(clf_avgw2v, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,retur
n_train_score=True)
```

```
model_avgw2v.fit(X_tr_avgw2v, y_train_avgw2v)



print(model_avgw2v.best_estimator_)
print(model_avgw2v.score(X_te_avgw2v, y_test_avgw2v))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   37.9s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed: 11.2min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 47.9min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 92.6min finished
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=6, max_features='auto',
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)
0.6784105227396852
```

In [204]:

```
train_auc= model_avgw2v.cv_results_['mean_train_score']
cv_auc = model_avgw2v.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

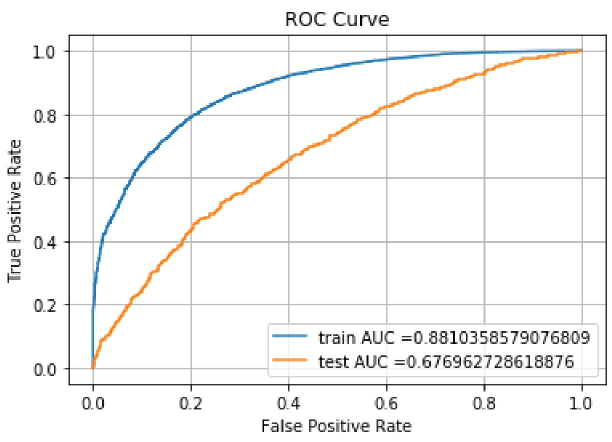
In [205]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [207]:

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

clf_avgw2v = RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=6, max_features='auto',
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)

clf_avgw2v.fit(X_tr_avgw2v, y_train_avgw2v)


y_train_pred = clf_avgw2v.predict_proba(X_tr_avgw2v)[:,1]
y_test_pred = clf_avgw2v.predict_proba(X_te_avgw2v)[:,1]




train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_avgw2v, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_avgw2v, 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 [208]:

```
# 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 [209]:

```
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) 0.6332099590537884 for threshold 0.515
```

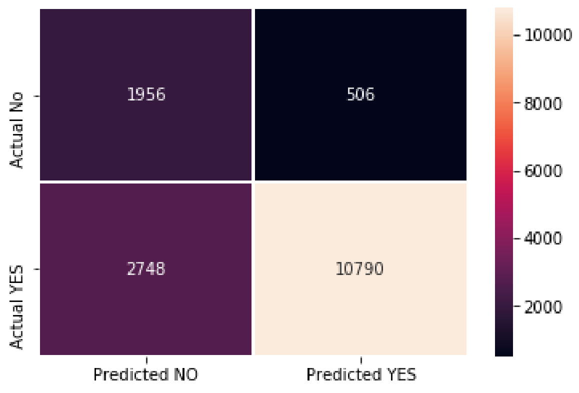
In [210]:

```
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 [211]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train_avgw2v, predict_with_best_t(y_train_pred, best_t))
```

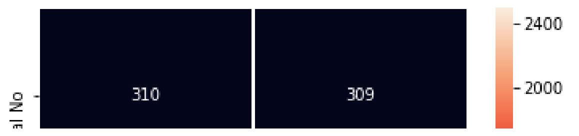
```
Train confusion matrix
```

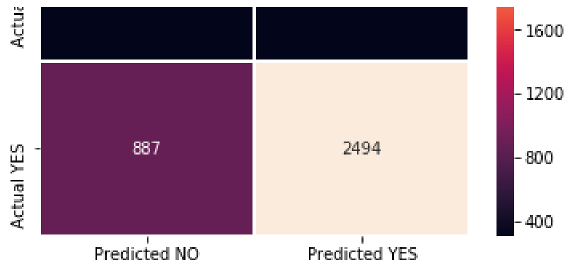


In [212]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test_avgw2v, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```





3.4.4 Applying Random Forests on TFIDF W2V, SET 4

In [220]:

```
# 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_tfidfw2v = hstack((train_20_state_encoded_0,train_20_state_encoded_1, train_20_teacher_prefix_enco
ded_0,train_20_teacher_prefix_encoded_1, train_20_grade_encoded_0,train_20_grade_encoded_1, train_20_ca
tegories_encoded_0,train_20_categories_encoded_1, train_20_subcategories_encoded_0,train_20_subcategori
es_encoded_1, X_train_20_price_scaler, X_train_20_posted_project_scaler, essay_tfidf_w2v_train, title_t
fidf_w2v_train)).tocsr()

X_te_tfidfw2v = hstack((test_20_state_encoded_0,test_20_state_encoded_1, test_20_teacher_prefix_encoded
_0,test_20_teacher_prefix_encoded_1, test_20_grade_encoded_0,test_20_grade_encoded_1, test_20_categorie
s_encoded_0,test_20_categories_encoded_1, test_20_subcategories_encoded_0,test_20_subcategories_encoded
_1, X_test_20_price_scaler, X_test_20_posted_project_scaler, essay_tfidf_w2v_test, title_tfidf_w2v_test
)).tocsr()

y_train_tfidfw2v = y_train_20

y_test_tfidfw2v = y_test_20

print("Final Data matrix")
print(X_tr_tfidfw2v.shape, y_train_tfidfw2v.shape)

print(X_te_tfidfw2v.shape, y_test_tfidfw2v.shape)
print("="*100)
```

```
Final Data matrix
(16000, 612) (16000,)
(4000, 612) (4000,)
====================================================================================================
```

3.4.4.1 Hyperparameter Tuning

In [221]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt



tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_tfidfw2v = RandomForestClassifier(class_weight='balanced', n_jobs=-1)

#Using GridSearchCV
model_tfidfw2v = GridSearchCV(clf_tfidfw2v, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,r
eturn_train_score=True)
model_tfidfw2v.fit(X_tr_tfidfw2v, y_train_tfidfw2v)



print(model_tfidfw2v.best_estimator_)
print(model_tfidfw2v.score(X_te_tfidfw2v, y_test_tfidfw2v))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   31.9s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  9.9min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 42.8min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 186.5min finished
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=6, max_features='auto',
                       max_leaf_nodes=None, min_impurity_decrease=0.0,
                       min_impurity_split=None, min_samples_leaf=1,
                       min_samples_split=2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000, n_jobs=-1, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)
0.6779413036549873
```

In [222]:

```
train_auc= model_tfidfw2v.cv_results_['mean_train_score']
cv_auc = model_tfidfw2v.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

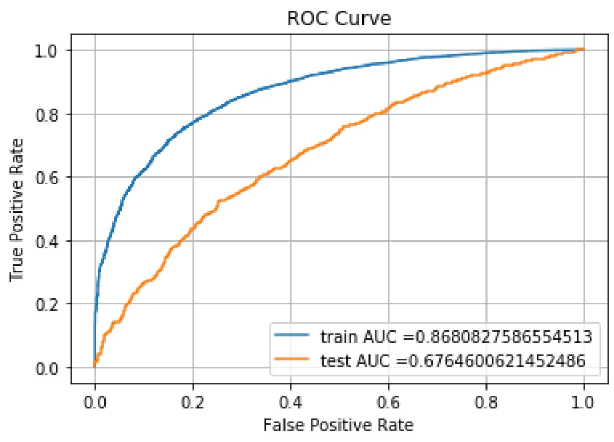
In [223]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [224]:

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

clf_tfidfw2v = RandomForestClassifier(bootstrap=True, class_weight='balanced',
                        criterion='gini', max_depth=6, max_features='auto',
                        max_leaf_nodes=None, min_impurity_decrease=0.0,
                        min_impurity_split=None, min_samples_leaf=1,
                        min_samples_split=2, min_weight_fraction_leaf=0.0,
                        n_estimators=1000, n_jobs=-1, oob_score=False,
                        random_state=None, verbose=0, warm_start=False)

clf_tfidfw2v.fit(X_tr_tfidfw2v, y_train_tfidfw2v)


y_train_pred = clf_tfidfw2v.predict_proba(X_tr_tfidfw2v)[:,1]
y_test_pred = clf_tfidfw2v.predict_proba(X_te_tfidfw2v)[:,1]




train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_tfidfw2v, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidfw2v, 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 [225]:

```
# 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 [226]:

```
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) 0.616878098283149 for threshold 0.512
```

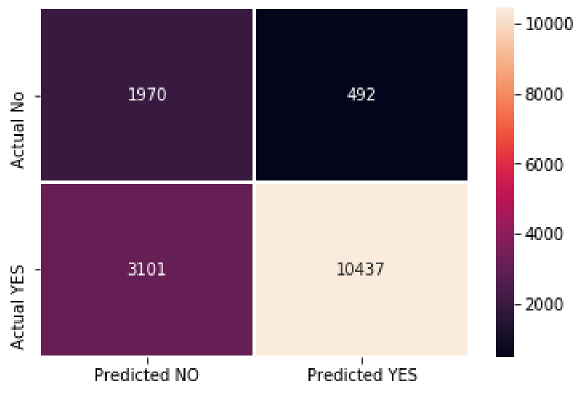
In [227]:

```
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 [228]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train_tfidfw2v, predict_with_best_t(y_train_pred, best_t))
```

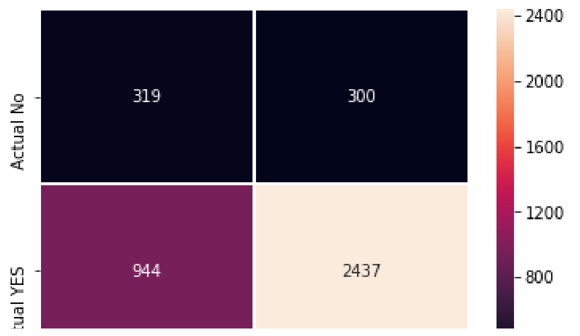
```
Train confusion matrix
```



In [229]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test_tfidfw2v, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



3.5 Applying GBDT

Apply GBDT 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

3.5.1 Applying XGBOOST on BOW, SET 1

In [0]:

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

In [234]:

```
# 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_bow = hstack((train_50_state_encoded_0,train_50_state_encoded_1, train_50_teacher_prefix_encoded_0
,train_50_teacher_prefix_encoded_1, train_50_grade_encoded_0,train_50_grade_encoded_1, train_50_categor
ies_encoded_0,train_50_categories_encoded_1, train_50_subcategories_encoded_0,train_50_subcategories_en
coded_1, X_train_50_price_scaler, X_train_50_posted_project_scaler, X_train_essay_bow, X_train_title_bo
w)).tocsr()

X_te_bow = hstack((test_50_state_encoded_0,test_50_state_encoded_1, test_50_teacher_prefix_encoded_0,te
st_50_teacher_prefix_encoded_1, test_50_grade_encoded_0,test_50_grade_encoded_1, test_50_categories_enc
oded_0,test_50_categories_encoded_1, test_50_subcategories_encoded_0,test_50_subcategories_encoded_1, X
_test_50_price_scaler, X_test_50_posted_project_scaler, X_test_essay_bow, X_test_title_bow)).tocsr()

y_train_bow = y_train_50

y_test_bow = y_test_50

print("Final Data matrix")
print(X_tr_bow.shape, y_train_bow.shape)

print(X_te_bow.shape, y_test_bow.shape)
print("="*100)
```

```
Final Data matrix
(40000, 14244) (40000,)
(10000, 14244) (10000,)
====================================================================================================
```

3.5.1.1 Hyperparameter Tuning

In [241]:

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
import matplotlib.pyplot as plt



tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_bow = XGBClassifier(class_weight='balanced', n_jobs=-1)

#Using GridSearchCV
model_bow = GridSearchCV(clf_bow, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,return_trai
n_score=True)
model_bow.fit(X_tr_bow, y_train_bow)
```

```
print(model_bow.best_estimator_)
print(model_bow.score(X_te_bow, y_test_bow))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:  1.9min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed: 38.1min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 299.9min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 437.1min finished
```

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=2,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=-1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
0.724836118049106
```

In [246]:

```
train_auc= model_bow.cv_results_['mean_train_score']
cv_auc = model_bow.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

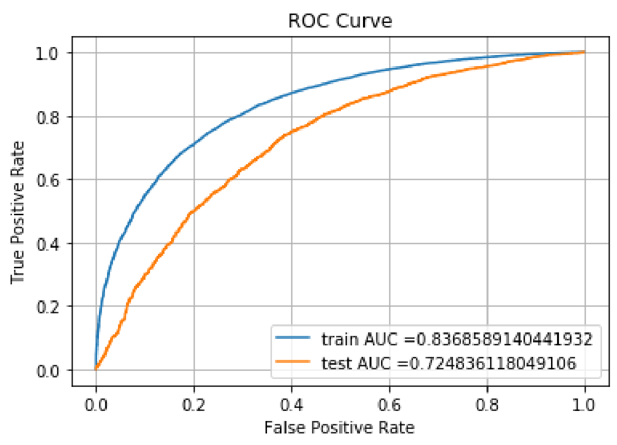
In [245]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [247]:

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

clf_bow = XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=2,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=-1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)

clf_bow.fit(X_tr_bow, y_train_bow)


y_train_pred = clf_bow.predict_proba(X_tr_bow)[:,1]
y_test_pred = clf_bow.predict_proba(X_te_bow)[:,1]




train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_bow, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_bow, 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 [248]:

```
# 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 [249]:

```
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) 0.5742646129009629 for threshold 0.826
```

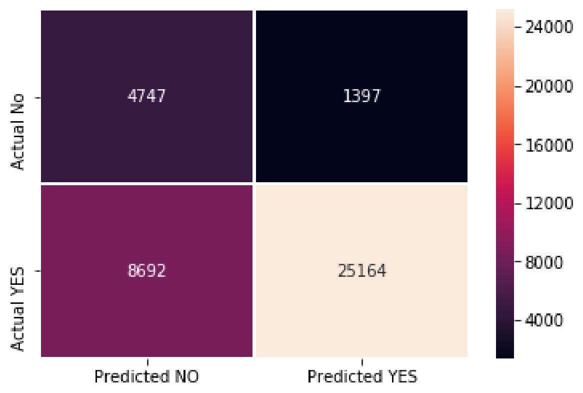
In [250]:

```
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 [251]:

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

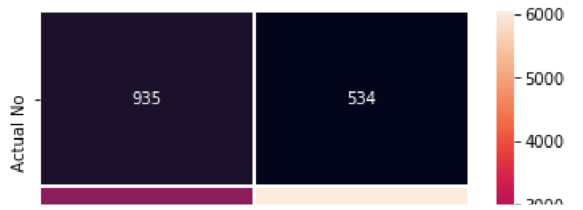
```
Train confusion matrix
```

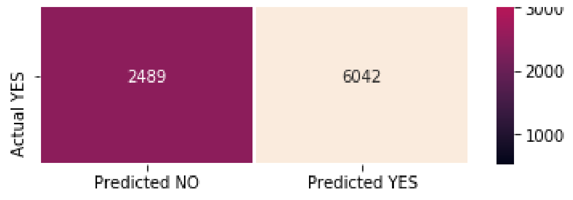


In [252]:

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

```
Test confusion matrix
```





3.5.2 Applying XGBOOST on TFIDF, SET 2

In [0]:

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

In [253]:

```
# 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_tfidf = hstack((train_50_state_encoded_0,train_50_state_encoded_1, train_50_teacher_prefix_encoded
_0,train_50_teacher_prefix_encoded_1, train_50_grade_encoded_0,train_50_grade_encoded_1, train_50_categ
ories_encoded_0,train_50_categories_encoded_1, train_50_subcategories_encoded_0,train_50_subcategories_
encoded_1, X_train_50_price_scaler, X_train_50_posted_project_scaler, X_train_essay_tfidf, X_train_titl
e_tfidf)).tocsr()

X_te_tfidf = hstack((test_50_state_encoded_0,test_50_state_encoded_1, test_50_teacher_prefix_encoded_0,
test_50_teacher_prefix_encoded_1, test_50_grade_encoded_0,test_50_grade_encoded_1, test_50_categories_e
ncoded_0,test_50_categories_encoded_1, test_50_subcategories_encoded_0,test_50_subcategories_encoded_1,
X_test_50_price_scaler, X_test_50_posted_project_scaler, X_test_essay_tfidf, X_test_title_tfidf)).tocsr
()

y_train_tfidf = y_train_50

y_test_tfidf = y_test_50

print("Final Data matrix")
print(X_tr_tfidf.shape, y_train_tfidf.shape)

print(X_te_tfidf.shape, y_test_tfidf.shape)
print("="*100)
```

```
Final Data matrix
(40000, 14244) (40000,)
(10000, 14244) (10000,)
====================================================================================================
```

3.5.2.1 Hyperparameter Tuning

In [258]:

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier
import matplotlib.pyplot as plt



tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_tfidf = XGBClassifier(class_weight='balanced', n_jobs=-1)


#Using GridSearchCV
model_tfidf = GridSearchCV(clf_tfidf, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,return_
train_score=True)
model_tfidf.fit(X_tr_tfidf, y_train_tfidf)
```

```
print(model_tfidf.best_estimator_)
print(model_tfidf.score(X_te_tfidf, y_test_tfidf))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fitsFitting 3 folds for each of 72 candidates,
totalling 216 fitsFitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:  4.1min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed: 84.9min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed: 84.9min
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed: 84.9min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 435.6min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 741.6min finished
```

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=2,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=-1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
0.7181712010312129
```

In [259]:

```
train_auc= model_tfidf.cv_results_['mean_train_score']
cv_auc = model_tfidf.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

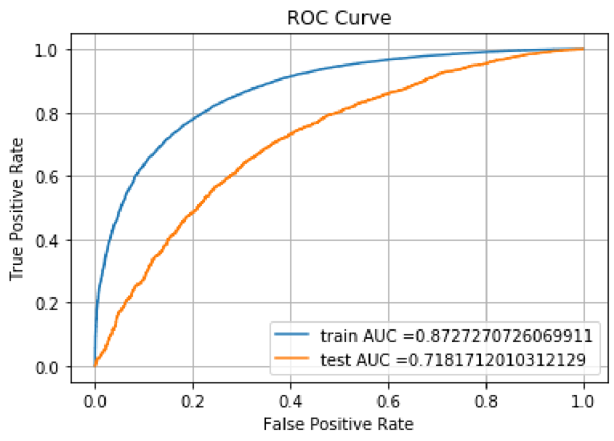
In [260]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [261]:

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

clf_tfidf = XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=2,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=-1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)

clf_tfidf.fit(X_tr_tfidf, y_train_tfidf)


y_train_pred = clf_tfidf.predict_proba(X_tr_tfidf)[:,1]
y_test_pred = clf_tfidf.predict_proba(X_te_tfidf)[:,1]




train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_tfidf, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidf, 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 [262]:

```
# 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 [263]:

```
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) 0.6234177299167799 for threshold 0.82
```

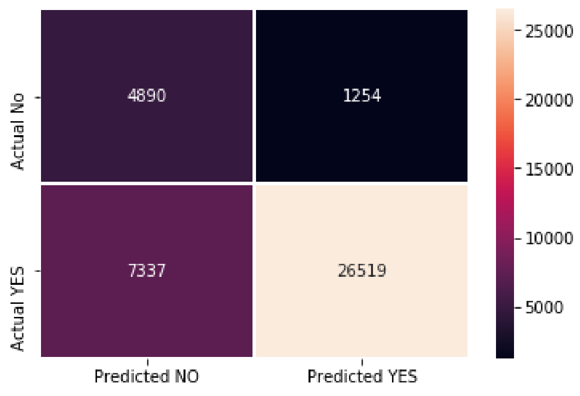
In [264]:

```
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 [265]:

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

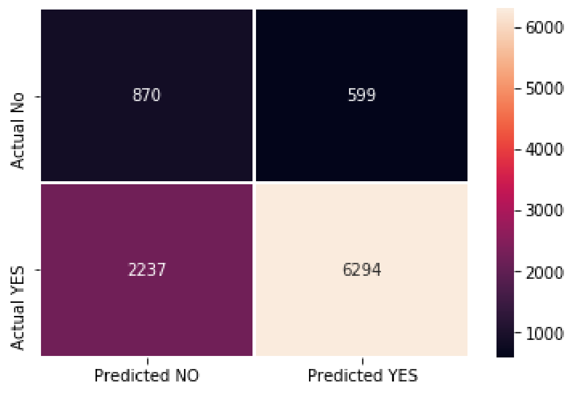
```
Train confusion matrix
```



In [266]:

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

```
Test confusion matrix
```



3.5.3 Applying lightGBM on AVG W2V, SET 3

In [267]:

```
# 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_avgw2v = hstack((train_20_state_encoded_0,train_20_state_encoded_1, train_20_teacher_prefix_encode
d_0,train_20_teacher_prefix_encoded_1, train_20_grade_encoded_0,train_20_grade_encoded_1, train_20_cate
gories_encoded_0,train_20_categories_encoded_1, train_20_subcategories_encoded_0,train_20_subcategories
_encoded_1, X_train_20_price_scaler, X_train_20_posted_project_scaler, avg_w2v_essay_train, avg_w2v_tit
le_train)).tocsr()

X_te_avgw2v = hstack((test_20_state_encoded_0,test_20_state_encoded_1, test_20_teacher_prefix_encoded_0
,test_20_teacher_prefix_encoded_1, test_20_grade_encoded_0,test_20_grade_encoded_1, test_20_categories_
encoded_0,test_20_categories_encoded_1, test_20_subcategories_encoded_0,test_20_subcategories_encoded_1
, X_test_20_price_scaler, X_test_20_posted_project_scaler, avg_w2v_essay_test, avg_w2v_title_test)).toc
sr()

y_train_avgw2v = y_train_20

y_test_avgw2v = y_test_20

print("Final Data matrix")
print(X_tr_avgw2v.shape, y_train_avgw2v.shape)

print(X_te_avgw2v.shape, y_test_avgw2v.shape)
print("="*100)
```

```
Final Data matrix
(16000, 612) (16000,)
(4000, 612) (4000,)
====================================================================================================
```

3.5.3.1 Hyperparameter Tuning

In [269]:

```
from sklearn.model_selection import GridSearchCV
from lightgbm import LGBMClassifier
import matplotlib.pyplot as plt



tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_avgw2v = LGBMClassifier(class_weight='balanced', n_jobs=-1)

#Using GridSearchCV
model_avgw2v = GridSearchCV(clf_avgw2v, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,retur
n_train_score=True)
model_avgw2v.fit(X_tr_avgw2v, y_train_avgw2v)
```

```
print(model_avgw2v.best_estimator_)
print(model_avgw2v.score(X_te_avgw2v, y_test_avgw2v))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   34.4s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  7.1min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 41.1min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 80.9min finished
```

```
LGBMClassifier(boosting_type='gbdt', class_weight='balanced',
               colsample_bytree=1.0, importance_type='split', learning_rate=0.1,
               max_depth=2, min_child_samples=20, min_child_weight=0.001,
               min_split_gain=0.0, n_estimators=150, n_jobs=-1, num_leaves=31,
               objective=None, random_state=None, reg_alpha=0.0, reg_lambda=0.0,
               silent=True, subsample=1.0, subsample_for_bin=200000,
               subsample_freq=0)
0.6803218021070899
```

In [270]:

```
train_auc= model_avgw2v.cv_results_['mean_train_score']
cv_auc = model_avgw2v.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

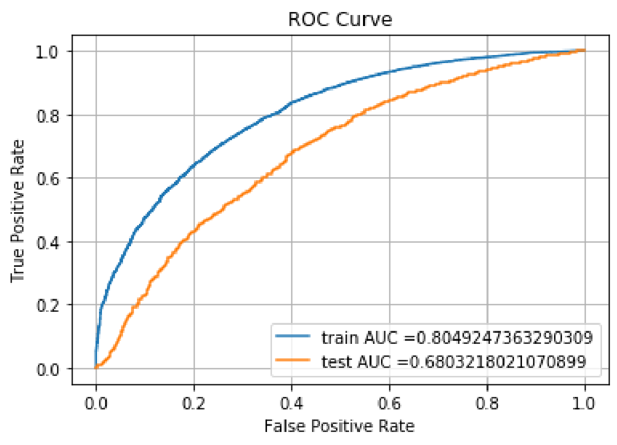
In [271]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [272]:

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

clf_avgw2v = LGBMClassifier(boosting_type='gbdt', class_weight='balanced',
               colsample_bytree=1.0, importance_type='split', learning_rate=0.1,
               max_depth=2, min_child_samples=20, min_child_weight=0.001,
               min_split_gain=0.0, n_estimators=150, n_jobs=-1, num_leaves=31,
               objective=None, random_state=None, reg_alpha=0.0, reg_lambda=0.0,
               silent=True, subsample=1.0, subsample_for_bin=200000,
               subsample_freq=0)

clf_avgw2v.fit(X_tr_avgw2v, y_train_avgw2v)


y_train_pred = clf_avgw2v.predict_proba(X_tr_avgw2v)[:,1]
y_test_pred = clf_avgw2v.predict_proba(X_te_avgw2v)[:,1]




train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_avgw2v, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_avgw2v, 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 [273]:

```
# 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 [274]:

```
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) 0.5250205847151185 for threshold 0.499
```

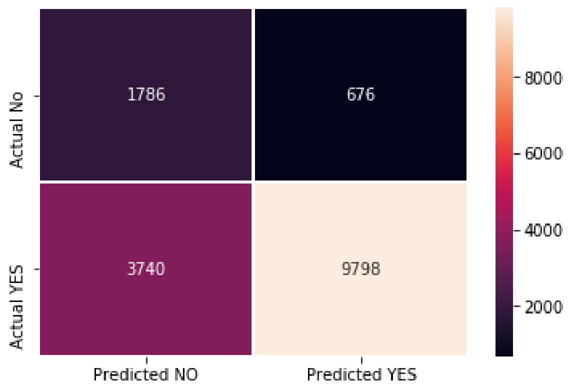
In [275]:

```
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 [276]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train_avgw2v, predict_with_best_t(y_train_pred, best_t))
```

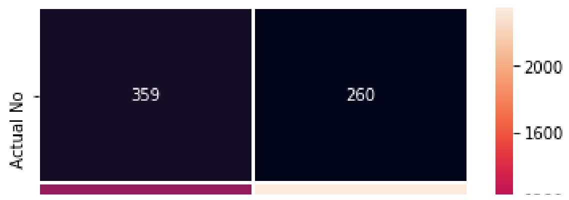
```
Train confusion matrix
```

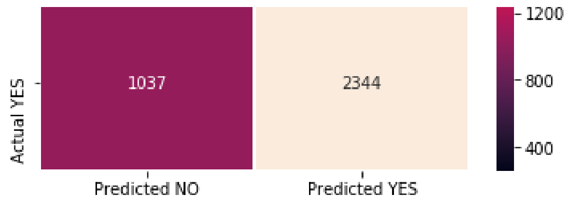


In [277]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test_avgw2v, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```





3.5.4 Applying lightGBM on TFIDF W2V, SET 4

In [0]:

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

In [278]:

```
# 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_tfidfw2v = hstack((train_20_state_encoded_0,train_20_state_encoded_1, train_20_teacher_prefix_enco
ded_0,train_20_teacher_prefix_encoded_1, train_20_grade_encoded_0,train_20_grade_encoded_1, train_20_ca
tegories_encoded_0,train_20_categories_encoded_1, train_20_subcategories_encoded_0,train_20_subcategori
es_encoded_1, X_train_20_price_scaler, X_train_20_posted_project_scaler, essay_tfidf_w2v_train, title_t
fidf_w2v_train)).tocsr()

X_te_tfidfw2v = hstack((test_20_state_encoded_0,test_20_state_encoded_1, test_20_teacher_prefix_encoded
_0,test_20_teacher_prefix_encoded_1, test_20_grade_encoded_0,test_20_grade_encoded_1, test_20_categorie
s_encoded_0,test_20_categories_encoded_1, test_20_subcategories_encoded_0,test_20_subcategories_encoded
_1, X_test_20_price_scaler, X_test_20_posted_project_scaler, essay_tfidf_w2v_test, title_tfidf_w2v_test
)).tocsr()

y_train_tfidfw2v = y_train_20

y_test_tfidfw2v = y_test_20

print("Final Data matrix")
print(X_tr_tfidfw2v.shape, y_train_tfidfw2v.shape)

print(X_te_tfidfw2v.shape, y_test_tfidfw2v.shape)
print("="*100)
```

```
Final Data matrix
(16000, 612) (16000,)
(4000, 612) (4000,)
====================================================================================================
```

3.5.4.1 Hyperparameter Tuning

In [279]:

```
from sklearn.model_selection import GridSearchCV
from lightgbm import LGBMClassifier
import matplotlib.pyplot as plt



tuned_parameters = {'max_depth':[2,3,4,5,6,7,8,9,10],'n_estimators':[10,50,100,150,200,300,500,1000]}

clf_tfidfw2v = LGBMClassifier(class_weight='balanced', n_jobs=-1)

#Using GridSearchCV
model_tfidfw2v = GridSearchCV(clf_tfidfw2v, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,r
eturn_train_score=True)
model_tfidfw2v.fit(X_tr_tfidfw2v, y_train_tfidfw2v)



print(model_tfidfw2v.best_estimator_)
print(model_tfidfw2v.score(X_te_tfidfw2v, y_test_tfidfw2v))
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   29.3s
[Parallel(n_jobs=-1)]: Done  64 tasks      | elapsed:  7.3min
[Parallel(n_jobs=-1)]: Done 154 tasks      | elapsed: 43.0min
[Parallel(n_jobs=-1)]: Done 216 out of 216 | elapsed: 81.0min finished
```

```
LGBMClassifier(boosting_type='gbdt', class_weight='balanced',
               colsample_bytree=1.0, importance_type='split', learning_rate=0.1,
               max_depth=2, min_child_samples=20, min_child_weight=0.001,
               min_split_gain=0.0, n_estimators=150, n_jobs=-1, num_leaves=31,
               objective=None, random_state=None, reg_alpha=0.0, reg_lambda=0.0,
               silent=True, subsample=1.0, subsample_for_bin=200000,
               subsample_freq=0)
0.6791781403156192
```

In [280]:

```
train_auc= model_tfidfw2v.cv_results_['mean_train_score']
cv_auc = model_tfidfw2v.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
n_estimators = tuned_parameters['n_estimators']
print(n_estimators)
```

```
[2, 3, 4, 5, 6, 7, 8, 9, 10]
[10, 50, 100, 150, 200, 300, 500, 1000]
```

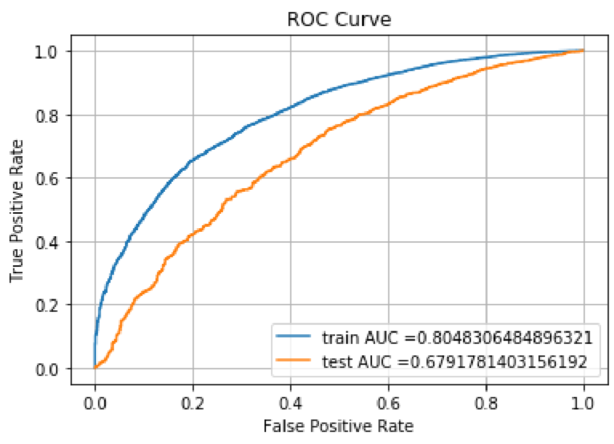
In [281]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=n_estimators,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=n_estimators,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

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

In [282]:

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

clf_tfidfw2v = LGBMClassifier(boosting_type='gbdt', class_weight='balanced',
               colsample_bytree=1.0, importance_type='split', learning_rate=0.1,
               max_depth=2, min_child_samples=20, min_child_weight=0.001,
               min_split_gain=0.0, n_estimators=150, n_jobs=-1, num_leaves=31,
               objective=None, random_state=None, reg_alpha=0.0, reg_lambda=0.0,
               silent=True, subsample=1.0, subsample_for_bin=200000,
               subsample_freq=0)

clf_tfidfw2v.fit(X_tr_tfidfw2v, y_train_tfidfw2v)


y_train_pred = clf_tfidfw2v.predict_proba(X_tr_tfidfw2v)[:,1]
y_test_pred = clf_tfidfw2v.predict_proba(X_te_tfidfw2v)[:,1]




train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_tfidfw2v, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidfw2v, 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 [283]:

```
# 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 [284]:

```
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) 0.5271050683942986 for threshold 0.504
```

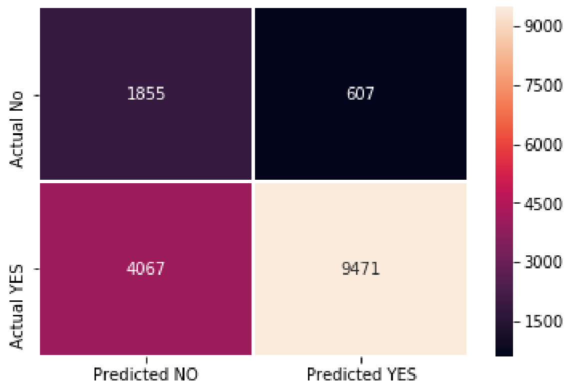
In [285]:

```
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 [286]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train_tfidfw2v, predict_with_best_t(y_train_pred, best_t))
```

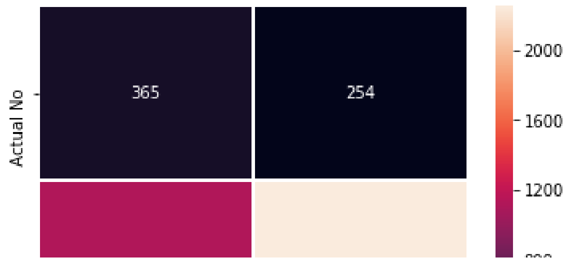
```
Train confusion matrix
```

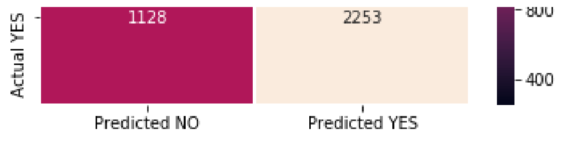


In [287]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test_tfidfw2v, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```





4. Conclusion

Random Forest

In [289]:

```
# 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(max_depth)", "Hyperparameter(n_estimators)", "A
UC"]

x.add_row(["BOW", "Auto", 10, 1000, 0.70387])
x.add_row(["TFIDF", "Auto", 10, 1000, 0.70742])
x.add_row(["ACG W2V", "Auto", 6, 1000, 0.67696])
x.add_row(["TFIDF W2V", "Auto", 6, 1000, 0.67646])



print(x)
```

```
+------------+-------+---------------------------+------------------------------+---------+
| Vectorizer | Model | Hyperparameter(max_depth) | Hyperparameter(n_estimators) |   AUC   |
+------------+-------+---------------------------+------------------------------+---------+
|    BOW     |  Auto |             10            |             1000             | 0.70387 |
|   TFIDF    |  Auto |             10            |             1000             | 0.70742 |
|  ACG W2V   |  Auto |             6             |             1000             | 0.67696 |
| TFIDF W2V  |  Auto |             6             |             1000             | 0.67646 |
+------------+-------+---------------------------+------------------------------+---------+
```

GBDT

In [291]:

```
# 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(max_depth)", "Hyperparameter(n_estimators)", "A
UC"]


x.add_row(["BOW", "Auto", 2, 1000, 0.72483])
x.add_row(["TFIDF", "Auto", 2, 1000, 0.71817])
x.add_row(["ACG W2V", "Auto", 2, 150, 0.68032])
x.add_row(["TFIDF W2V", "Auto", 2, 150, 0.67917])



print(x)
```

```
+------------+-------+---------------------------+------------------------------+---------+
| Vectorizer | Model | Hyperparameter(max_depth) | Hyperparameter(n_estimators) |   AUC   |
+------------+-------+---------------------------+------------------------------+---------+
|    BOW     |  Auto |             2             |             1000             | 0.72483 |
|   TFIDF    |  Auto |             2             |             1000             | 0.71817 |
|  ACG W2V   |  Auto |             2             |             150              | 0.68032 |
| TFIDF W2V  |  Auto |             2              |             150              | 0.67917 |
+------------+-------+----------------------------+------------------------------+---------+
```