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	Desc
`project_id`	A unique identifier for the proposed project. **Example:** `p0`
`project_title`	Title of the project. **Exa** • `Art Will Make You H` • `First Grad`
`project_grade_category`	Grade level of students for which the project is targeted. One of the fo enumerated v • `Grades P` • `Grade` • `Grade` • `Grades`

Feature	Desc
project_subject_categories	One or more (comma-separated) subject categories for the project fr following enumerated list of v • Applied Lea • Care & H • Health & S • History & C • Literacy & Lan • Math & Sc • Music & The • Special • W **Exar** • Music & The • Literacy & Language, Math & Sc
school_state	State where school is located (Two-letter U.S. posta (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_c **Exampl**
project_subject_subcategories	One or more (comma-separated) subject subcategories for the p **Exar** • Lit • Literature & Writing, Social Sci
project_resource_summary	An explanation of the resources needed for the project. **Exa** • My students need hands on literacy materials to ma sensory needs!<
project_essay_1	First application
project_essay_2	Second application
project_essay_3	Third application
project_essay_4	Fourth application
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2016-(12:43:5
teacher_id	A unique identifier for the teacher of the proposed project. **Exa** bdf8baa8fedef6bfeec7ae4ff1c
teacher_prefix	Teacher's title. One of the following enumerated v • • • • • • Tea
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same te **Examp**

[*] 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`

Feature	Description
`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
```

```
C:\Users\MONIKA KUMARI\Anaconda3\lib\site-packages\gensim\utils.py:1197: Use
rWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:

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

In [3]:

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

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

In [4]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[4]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# 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", '
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# 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", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

1.3 Text preprocessing

In [7]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [8]:

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

Out[8]:

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

In [9]:

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

In [10]:

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

```
My students are English learners that are working on English as their second
or third languages. We are a melting pot of refugees, immigrants, and native
-born Americans bringing the gift of language to our school. \r\n\r\n We hav
e over 24 languages represented in our English Learner program with students
at every level of mastery.  We also have over 40 countries represented with
the families within our school.  Each student brings a wealth of knowledge a
nd experiences to us that open our eyes to new cultures, beliefs, and respec
t.\"The limits of your language are the limits of your world.\"-Ludwig Wittg
enstein  Our English learner's have a strong support system at home that beg
s for more resources.  Many times our parents are learning to read and speak
English along side of their children.  Sometimes this creates barriers for p
arents to be able to help their child learn phonetics, letter recognition, a
nd other reading skills.\r\n\r\nBy providing these dvd's and players, studen
ts are able to continue their mastery of the English language even if no one
at home is able to assist.  All families with students within the Level 1 pr
oficiency status, will be a offered to be a part of this program.  These edu
cational videos will be specially chosen by the English Learner Teacher and
will be sent home regularly to watch.  The videos are to help the child deve
lop early reading skills.\r\n\r\nParents that do not have access to a dvd pl
ayer will have the opportunity to check out a dvd player to use for the yea
r.  The plan is to use these videos and educational dvd's for the years to c
ome for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year a
ll love learning, at least most of the time. At our school, 97.3% of the stu
dents receive free or reduced price lunch. Of the 560 students, 97.3% are mi
nority students. \r\nThe school has a vibrant community that loves to get to
gether and celebrate. Around Halloween there is a whole school parade to sho
w off the beautiful costumes that students wear. On Cinco de Mayo we put on
a big festival with crafts made by the students, dances, and games. At the e
nd of the year the school hosts a carnival to celebrate the hard work put in
during the school year, with a dunk tank being the most popular activity.My
students will use these five brightly colored Hokki stools in place of regul
ar, stationary, 4-legged chairs. As I will only have a total of ten in the c
lassroom and not enough for each student to have an individual one, they wil
l 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 th
em in place of chairs at my small group tables during math and reading time
s. The rest of the day they will be used by the students who need the highes
t 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. Whe
n the students are sitting in group with me on the Hokki Stools, they are al
ways moving, but at the same time doing their work. Anytime the students get
to pick where they can sit, the Hokki Stools are the first to be taken. Ther
e are always students who head over to the kidney table to get one of the st
ools who are disappointed as there are not enough of them. \r\n\r\nWe ask a
```

```
lot of students to sit for 7 hours a day. The Hokki stools will be a comprom
ise 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 allo
wing them to activate their core muscles for balance while they sit. For man
y of my students, these chairs will take away the barrier that exists in sch
ools for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment wit
h plain walls, rows of desks, and a teacher in front of the room? A typical
day in our room is nothing like that. I work hard to create a warm inviting
themed room for my students look forward to coming to each day.\r\n\r\nMy cl
ass is made up of 28 wonderfully unique boys and girls of mixed races in Ark
ansas.\r\nThey attend a Title I school, which means there is a high enough p
ercentage of free and reduced-price lunch to qualify. Our school is an \"ope
n classroom\" concept, which is very unique as there are no walls separating
the classrooms. These 9 and 10 year-old students are very eager learners; th
ey are like sponges, absorbing all the information and experiences and keep
on wanting more.With these resources such as the comfy red throw pillows and
the whimsical nautical hanging decor and the blue fish nets, I will be able
to help create the mood in our classroom setting to be one of a themed nauti
cal environment. Creating a classroom environment is very important in the s
uccess in each and every child's education. The nautical photo props will be
used with each child as they step foot into our classroom for the first time
on Meet the Teacher evening. I'll take pictures of each child with them, hav
e them developed, and then hung in our classroom ready for their first day o
f 4th grade.  This kind gesture will set the tone before even the first day
of school! The nautical thank you cards will be used throughout the year by
the students as they create thank you cards to their team groups.\r\n\r\nYou
r generous donations will help me to help make our classroom a fun, invitin
g, 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 helpin
g 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 la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations. \r\n\r\nThe materials we have are the ones I seek out for my stu
dents. I teach in a Title I school where most of the students receive free o
r reduced price lunch.  Despite their disabilities and limitations, my stude
nts love coming to school and come eager to learn and explore.Have you ever
felt like you had ants in your pants and you needed to groove and move as yo
u were in a meeting? This is how my kids feel all the time. The want to be a
ble 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 d
on't want to sit and do worksheets. They want to learn to count by jumping a
nd playing. Physical engagement is the key to our success. The number toss a
nd color and shape mats can make that happen. My students will forget they a
re doing work and just have the fun a 6 year old deserves.nannan
==================================================
```

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

In [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 la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations. \r\n\r\nThe materials we have are the ones I seek out for my stu
dents. I teach in a Title I school where most of the students receive free o
r reduced price lunch.  Despite their disabilities and limitations, my stude
nts love coming to school and come eager to learn and explore.Have you ever
felt like you had ants in your pants and you needed to groove and move as yo
u were in a meeting? This is how my kids feel all the time. The want to be a
ble 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 d
o 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 [13]:

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

```
My kindergarten students have varied disabilities ranging from speech and la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations.     The materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or re
duced price lunch.  Despite their disabilities and limitations, my students
love coming to school and come eager to learn and explore.Have you ever felt
like you had ants in your pants and you needed to groove and move as you wer
e in a meeting? This is how my kids feel all the time. The want to be able t
o 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.   They also want to learn through games, my kids do not w
ant to sit and do worksheets. They want to learn to count by jumping and pla
ying. Physical engagement is the key to our success. The number toss and col
or and shape mats can make that happen. My students will forget they are doi
ng 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 la
nguage delays cognitive delays gross fine motor delays to autism They are ea
ger beavers and always strive to work their hardest working past their limit
ations The materials we have are the ones I seek out for my students I teach
in a Title I school where most of the students receive free or reduced price
lunch Despite their disabilities and limitations my students love coming to
school and come eager to learn and explore Have you ever felt like you had a
nts 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 le
arn 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
They also want to learn through games my kids do not want to sit and do work
sheets They want to learn to count by jumping and playing Physical engagemen
t is the key to our success The number toss and color and shape mats can mak
e 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            '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 [02:45<00:00, 660.79i
t/s]
```

In [17]:

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

Out[17]:

```
'my kindergarten students varied disabilities ranging speech language delays
cognitive delays gross fine motor delays autism they eager beavers always st
rive work hardest working past limitations the materials ones i seek student
s i teach title i school students receive free reduced price lunch despite d
isabilities limitations students love coming school come eager learn explore
have ever felt like ants pants needed groove move meeting this kids feel tim
e the want able move learn say wobble chairs answer i love develop core enha
nces gross motor turn fine motor skills they also want learn games kids not
want sit worksheets they want learn count jumping playing physical engagemen
t key success the number toss color shape mats make happen my students forge
t work fun 6 year old deserves nannan'
```

In [18]:

```
project_data['preprocessed_essays'] = 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_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

1.4 Preprocessing of `project_title`

In [19]:

```
# printing some random project title.
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 [20]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bars
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)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████████████████████████| 109248/109248 [00:07<00:00, 15067.02i
t/s]
```

In [21]:

```
#after preprocessing
preprocessed_titles[20000]
```

Out[21]:

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

In [22]:

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

Out[22]:

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

1.5 Preprocessing teacher_prefix

In [23]:

```
x = project_data['teacher_prefix'].replace(to_replace= np.nan, value= "mrs")
teacher_prefix_list = list(x.values)
preprocessed_teacher_prefix=[]
for l in tqdm (teacher_prefix_list):
    n = ""
    for e in l:
        e = e.replace('.', '')
        e = e.replace(',', '')
        n+= e
    preprocessed_teacher_prefix.append(n.lower().strip())

print(len(preprocessed_teacher_prefix))
```

```
100%|███████████████| 109248/109248 [00:00<00:00, 179291.26i
t/s]

109248
```

In [24]:

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

Out[24]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:2

1.6 Preprocessing project_grade_category

In [25]:

```
project_grade_category_list = list(project_data['project_grade_category'].values)
preprocessed_project_grade_category=[]
for l in tqdm (project_grade_category_list):
    n = ""
    for e in l:
        e = e.replace(' ', '_')
        e = e.replace('-', '_')
        n+= e
    preprocessed_project_grade_category.append(n.lower().strip())

print(len(preprocessed_project_grade_category))
```

```
100%|██████████████████████████████| 109248/109248 [00:02<00:00, 49603.92i
t/s]

109248
```

In [26]:

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

Out[26]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:2

1.7. Computing Sentiment Scores for preprocessed_essays

In [28]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
#nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()
neg= []
pos= []
neu= []
compound= []
for k in tqdm(project_data['preprocessed_essays'], position= 0, leave= True):
    a= sid.polarity_scores(k)['neg']
    b= sid.polarity_scores(k)['pos']
    c= sid.polarity_scores(k)['neu']
    d= sid.polarity_scores(k)['compound']
    neg.append(a)
    pos.append(b)
    neu.append(c)
    compound.append(d)
```

```
C:\Users\MONIKA KUMARI\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:
20: UserWarning:

The twython library has not been installed. Some functionality from the twit
ter package will not be available.

100%|██████████████████████████████████| 109248/109248 [35:57<00:00, 50.64i
t/s]
```

In [29]:

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

In [30]:

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

Out[30]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:2

2 rows × 22 columns

In [31]:

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

In [32]:

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

Out[32]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:2

2 rows × 24 columns

1.8 Preparing data for models

In [33]:

```
project_data.columns
```

Out[33]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'preprocessed_essays',
       'preprocessed_title', 'preprocessed_teacher_prefix',
       'preprocessed_project_grade_category', 'neg', 'pos', 'neu', 'compoun
d',
       'price', 'quantity'],
      dtype='object')
```

we are going to consider

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

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

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

1.9 Loading Data

In [34]:

```
project_data.to_csv('data.csv',index=False)
```

In [35]:

```
data = pd.read_csv('data.csv').iloc[0:50000]
data.shape
```

Out[35]:

```
(50000, 24)
```

In [36]:

```
data.head(2)
```

Out[36]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:2

2 rows × 24 columns

In [37]:

```
print(data.columns)
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'preprocessed_essays',
       'preprocessed_title', 'preprocessed_teacher_prefix',
       'preprocessed_project_grade_category', 'neg', 'pos', 'neu', 'compoun
d',
       'price', 'quantity'],
      dtype='object')
```

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

In [38]:

```
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)
X.head(1)
```

Out[38]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetir
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:

1 rows × 23 columns

In [39]:

```
y = data['project_is_approved']
y = y.to_frame() #changing y to dataframe
#y
```

In [40]:

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

# X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, shuffle=Flase)#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify= y, ranc
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=


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

```
(22445, 23) (22445, 1)
(11055, 23) (11055, 1)
(16500, 23) (16500, 1)
```

In [41]:

```
#value count in numpy array #https://stackoverflow.com/a/28663910
unique, counts = np.unique(y_train, return_counts=True) #counts the majority and minority c
dict(zip(unique, counts))
```

Out[41]:

```
{0: 3463, 1: 18982}
```

2.1 Response encoding Categorical data

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

In [44]:

```
X_train['target']= y_train['project_is_approved']
X_train.head(2)
```

Out[44]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_
34939	116687	p177921	bae76621cc538f4e6c0b84e1254d3f62	GA	2017-01-07
15327	40296	p030219	fc73d1ef2ebda6322470c071ee8159c7	MS	2016-11-11

2 rows × 24 columns

In [43]:

```
#X_train['clean_categories'].value_counts()
print(len(X_train['clean_categories'].value_counts()))
```

```
48
```

In [45]:

```
def train_resp_enc_0(X_train, cat):
    '''Calculating the number of unique values in category
    that belong to class 0 or project rejected.
    '''
    table = X_train[cat].unique()
    resp_0 = []
    #resp_1 = []
    for i in table:
        #print(i)
        c0 = (len(X_train.loc[(X_train[cat] == i) & (X_train['target']==0)]))#.append(resp_
        resp_0.append(c0)
    return resp_0

def train_resp_enc_1(X_train, cat):
    '''Calculating the number of unique values in category
    that belong to class 1 or project approved.
    '''
    table = X_train[cat].unique()
    resp_1 = []
    for i in table:
        c1 = (len(X_train.loc[(X_train[cat] == i) & (X_train['target']==1)]))#.append(resp_
        resp_1.append(c1)
    return resp_1
```

In [46]:

```
print(train_resp_enc_0(X_train, 'clean_categories'))
print(train_resp_enc_1(X_train, 'clean_categories'))
```

```
[381, 676, 68, 60, 153, 603, 69, 21, 43, 344, 152, 123, 148, 8, 10, 73, 31,
19, 69, 15, 33, 15, 45, 21, 70, 35, 20, 64, 20, 10, 4, 19, 4, 4, 11, 10, 0,
8, 0, 2, 1, 0, 0, 0, 0, 0, 0, 1]
[2645, 4228, 295, 295, 709, 2849, 422, 257, 222, 1796, 872, 696, 605, 65, 5
1, 380, 247, 97, 295, 108, 127, 48, 173, 251, 260, 123, 69, 297, 100, 42, 9,
146, 37, 26, 35, 33, 12, 18, 3, 9, 9, 5, 3, 2, 4, 3, 4, 0]
```

In [47]:

```
#res_0 = train_resp_enc_0(X_train, 'clean_categories')
#res_1 = train_resp_enc_1(X_train, 'clean_categories')

def enc_data(res_0,res_1, df, cat):
    '''Function to calculate the probability of each value in category for that belonging t
    and that belonging to class 1.
    '''
    enc_0 = []
    enc_1 = []
    #res_0 = train_resp_enc_0(X_train, cat)
    #res_1 = train_resp_enc_1(X_train, cat)
    for i in range(len(X_train[cat].unique())):
        enc_0.append(res_0[i]/(res_0[i]+res_1[i]))
        enc_1.append(res_1[i]/(res_0[i]+res_1[i]))
    #Arranging the probability values in a dataframe
    encoded_data= pd.DataFrame(list(zip(enc_0, enc_1)), columns=['class_0', 'class_1'], inc
    #encoded_data['state'] = (X_train[cat].unique())
    #encoded_data['class_0'] = encoded_data['class_0'].replace(np.nan, 0.5)
    #encoded_data['class_1'] = encoded_data['class_1'].replace(np.nan, 0.5)
    print("Shape of {} is {} ".format(cat,encoded_data.shape))
    return encoded_data
```

In [49]:

```
def response_encoding(df, cat):
    '''Using response coded values of train data in cv or test data.
    '''
    value_count = X_train[cat].value_counts()
    values = dict(value_count).keys()
    #print(values)
    feat_0=[]
    feat_1=[]
    for f, v in df[cat].iteritems():

        #print("f= ", f)
        #print("v= ",v)
        if v in values:
            #print("values= ", values)
            #print("v= ",v)
            feat_0.append(encoded_data['class_0'][v])
            feat_1.append(encoded_data['class_1'][v])
        else:
            feat_0.append(0.5)
            feat_1.append(0.5)
    #https://www.geeksforgeeks.org/create-a-pandas-dataframe-from-lists/
    respose_encoded= pd.DataFrame(list(zip(feat_0, feat_1)), columns=['class_0', 'class_1']
    #encoded_data['state'] = (X_train[cat].unique())
    respose_encoded['class_0'] = respose_encoded['class_0'].replace(np.nan, 0.5)
    respose_encoded['class_1'] = respose_encoded['class_1'].replace(np.nan, 0.5)
    print("Shape of {} is {} ".format(cat,respose_encoded.shape))
    return respose_encoded
```

2.1.1. Response encoding clean_categories for train data

In [48]:

```
res_0 = train_resp_enc_0(X_train, 'clean_categories')
res_1 = train_resp_enc_1(X_train, 'clean_categories')
encoded_data=enc_data(res_0,res_1,X_train, 'clean_categories')
encoded_data
```

```
Shape of clean_categories is (48, 2)
```

In [50]:

```
#response_encoding(X_train, 'clean_categories')
```

In [51]:

```
#tr_clean_cat_encd['class_0']
```

In [52]:

```
tr_clean_cat_encd = response_encoding(X_train, 'clean_categories')
cv_clean_cat_encd = response_encoding(X_cv, 'clean_categories')
te_clean_cat_encd = response_encoding(X_test, 'clean_categories')
```

```
Shape of clean_categories is (22445, 2)
Shape of clean_categories is (11055, 2)
Shape of clean_categories is (16500, 2)
```

2.1.2. Response encoding clean_subcategories for train data

In [53]:

```
res_0 = train_resp_enc_0(X_train, 'clean_subcategories')
res_1 = train_resp_enc_1(X_train, 'clean_subcategories')
encoded_data=enc_data(res_0,res_1, X_train, 'clean_subcategories')
encoded_data
```

```
Shape of clean_subcategories is (348, 2)
```

Out[53]:

	class_0	class_1
Literacy Mathematics	0.120139	0.879861
Literacy	0.115228	0.884772
FinancialLiteracy	0.193548	0.806452
AppliedSciences VisualArts	0.165468	0.834532
SpecialNeeds	0.177494	0.822506
Mathematics	0.163153	0.836847
EnvironmentalScience Literature_Writing	0.111111	0.888889
Literature_Writing	0.164627	0.835373
Literacy Literature_Writing	0.138381	0.861619
Civics_Government SocialSciences	0.117647	0.882353
History_Geography Literature_Writing	0.110092	0.889908
AppliedSciences College_CareerPrep	0.179775	0.820225
NutritionEducation	0.200000	0.800000
PerformingArts VisualArts	0.300000	0.700000
Literature_Writing SpecialNeeds	0.209738	0.790262
Literature_Writing Mathematics	0.132091	0.867909
Other	0.197674	0.802326
Civics_Government EnvironmentalScience	0.000000	1.000000
SpecialNeeds VisualArts	0.163934	0.836066
History_Geography	0.245098	0.754902
CharacterEducation Literature_Writing	0.212121	0.787879
Health_Wellness SpecialNeeds	0.107884	0.892116
Health_Wellness NutritionEducation	0.154762	0.845238
EnvironmentalScience Health_LifeScience	0.187192	0.812808
EarlyDevelopment Gym_Fitness	0.285714	0.714286
Gym_Fitness	0.185328	0.814672
AppliedSciences Mathematics	0.184862	0.815138
Mathematics SpecialNeeds	0.220339	0.779661
TeamSports	0.224880	0.775120
EnvironmentalScience History_Geography	0.027778	0.972222
...	...	...

	class_0	class_1
Other PerformingArts	1.000000	0.000000
College_CareerPrep History_Geography	0.500000	0.500000
EarlyDevelopment PerformingArts	0.000000	1.000000
CommunityService SocialSciences	0.500000	0.500000
FinancialLiteracy Health_LifeScience	0.000000	1.000000
EarlyDevelopment ForeignLanguages	0.000000	1.000000
Gym_Fitness Health_LifeScience	0.000000	1.000000
Mathematics TeamSports	0.500000	0.500000
Music Other	0.000000	1.000000
Civics_Government Health_LifeScience	0.000000	1.000000
EnvironmentalScience Extracurricular	0.000000	1.000000
College_CareerPrep ForeignLanguages	0.500000	0.500000
College_CareerPrep NutritionEducation	0.500000	0.500000
Gym_Fitness VisualArts	0.500000	0.500000
CharacterEducation ForeignLanguages	0.500000	0.500000
Health_Wellness PerformingArts	0.500000	0.500000
CharacterEducation Economics	1.000000	0.000000
Music TeamSports	0.000000	1.000000
EarlyDevelopment NutritionEducation	0.000000	1.000000
VisualArts Warmth Care_Hunger	1.000000	0.000000
AppliedSciences NutritionEducation	1.000000	0.000000
EnvironmentalScience Warmth Care_Hunger	0.000000	1.000000
EarlyDevelopment History_Geography	0.000000	1.000000
CommunityService ESL	1.000000	0.000000
EnvironmentalScience Music	0.000000	1.000000
Civics_Government ESL	0.000000	1.000000
Economics Literature_Writing	0.000000	1.000000
ParentInvolvement TeamSports	0.000000	1.000000
Extracurricular ForeignLanguages	0.000000	1.000000
ESL FinancialLiteracy	0.000000	1.000000

348 rows × 2 columns

In [54]:

```
tr_clean_subcat_encd = response_encoding(X_train, 'clean_subcategories')
cv_clean_subcat_encd = response_encoding(X_cv, 'clean_subcategories')
te_clean_subcat_encd = response_encoding(X_test, 'clean_subcategories')
```

```
Shape of clean_subcategories is (22445, 2)
Shape of clean_subcategories is (11055, 2)
Shape of clean_subcategories is (16500, 2)
```

2.1.3. Response encoding school_state for train data

In [55]:

```
res_0 = train_resp_enc_0(X_train, 'school_state')
res_1 = train_resp_enc_1(X_train, 'school_state')
encoded_data=enc_data(res_0,res_1, X_train,'school_state')
encoded_data
```

```
Shape of school_state is (51, 2)
```

Out[55]:

	class_0	class_1
GA	0.163772	0.836228
MS	0.143939	0.856061
ID	0.253623	0.746377
TX	0.180731	0.819269
MN	0.157895	0.842105
OH	0.128159	0.871841
NY	0.150096	0.849904
IN	0.149805	0.850195
AZ	0.155211	0.844789
SC	0.149830	0.850170
OK	0.137500	0.862500
NC	0.131068	0.868932
MI	0.166667	0.833333
WA	0.109442	0.890558
NV	0.150160	0.849840
NJ	0.183036	0.816964
CA	0.143807	0.856193
CT	0.121547	0.878453
MA	0.153509	0.846491
IL	0.176123	0.823877
SD	0.109375	0.890625
MO	0.153398	0.846602
WI	0.165354	0.834646
AR	0.188119	0.811881
FL	0.175494	0.824506
OR	0.189394	0.810606
PA	0.137405	0.862595
VA	0.151671	0.848329
MD	0.144295	0.855705
AL	0.146341	0.853659
LA	0.151064	0.848936

	class_0	class_1
TN	0.192771	0.807229
UT	0.153639	0.846361
KS	0.118881	0.881119
DC	0.173469	0.826531
WV	0.170000	0.830000
CO	0.169725	0.830275
HI	0.130081	0.869919
MT	0.229167	0.770833
KY	0.112319	0.887681
IA	0.193103	0.806897
NM	0.145833	0.854167
AK	0.115942	0.884058
ME	0.188679	0.811321
NE	0.112903	0.887097
WY	0.222222	0.777778
VT	0.153846	0.846154
DE	0.114286	0.885714
RI	0.178571	0.821429
NH	0.096774	0.903226
ND	0.115385	0.884615

In [56]:

```
tr_state_encd = response_encoding(X_train, 'school_state')
cv_state_encd = response_encoding(X_cv, 'school_state')
te_state_encd = response_encoding(X_test, 'school_state')
```

```
Shape of school_state is (22445, 2)
Shape of school_state is (11055, 2)
Shape of school_state is (16500, 2)
```

2.1.4. Response encoding preprocessed_project_grade_category for train data

In [57]:

```
res_0 = train_resp_enc_0(X_train, 'preprocessed_project_grade_category')
res_1 = train_resp_enc_1(X_train, 'preprocessed_project_grade_category')
encoded_data=enc_data(res_0,res_1, X_train,'preprocessed_project_grade_category')
encoded_data
```

```
Shape of preprocessed_project_grade_category is (4, 2)
```

Out[57]:

	class_0	class_1
grades_prek_2	0.151115	0.848885
grades_3_5	0.145975	0.854025
grades_9_12	0.177140	0.822860
grades_6_8	0.166618	0.833382

In [58]:

```
tr_grade_encd = response_encoding(X_train, 'preprocessed_project_grade_category')
cv_grade_encd = response_encoding(X_cv, 'preprocessed_project_grade_category')
te_grade_encd = response_encoding(X_test, 'preprocessed_project_grade_category')
```

```
Shape of preprocessed_project_grade_category is (22445, 2)
Shape of preprocessed_project_grade_category is (11055, 2)
Shape of preprocessed_project_grade_category is (16500, 2)
```

2.1.4. Response encoding preprocessed_teacher_prefix for train data

In [59]:

```
res_0 = train_resp_enc_0(X_train, 'preprocessed_teacher_prefix')
res_1 = train_resp_enc_1(X_train, 'preprocessed_teacher_prefix')
encoded_data=enc_data(res_0,res_1, X_train,'preprocessed_teacher_prefix')
encoded_data
```

```
Shape of preprocessed_teacher_prefix is (4, 2)
```

Out[59]:

	class_0	class_1
mrs	0.149673	0.850327
ms	0.155437	0.844563
teacher	0.190763	0.809237
mr	0.166982	0.833018

In [60]:

```
tr_prefix_encd = response_encoding(X_train, 'preprocessed_teacher_prefix')
cv_prefix_encd = response_encoding(X_cv, 'preprocessed_teacher_prefix')
te_prefix_encd = response_encoding(X_test, 'preprocessed_teacher_prefix')
```

```
Shape of preprocessed_teacher_prefix is (22445, 2)
Shape of preprocessed_teacher_prefix is (11055, 2)
Shape of preprocessed_teacher_prefix is (16500, 2)
```

2.2 Vectorizing Numerical features

In [63]:

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

In [64]:

```
data.head(2)
```

Out[64]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:4
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	FL	2016-10-25 09:2

2 rows × 24 columns

2.2.1. encoding numerical features: price

In [65]:

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

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

price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standard dev
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])

# Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_standardized_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_standardized_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
print(price_standardized_train.shape,y_train.shape)
print(price_standardized_cv.shape,y_cv.shape)
print(price_standardized_test.shape,y_test.shape)
```

```
Mean : 297.71562753397194, Standard deviation : 373.5890845438879
(22445, 1) (22445, 1)
(11055, 1) (11055, 1)
(16500, 1) (16500, 1)
```

In [66]:

```
price_standardized_train
```

Out[66]:

```
array([[-0.37665883],
       [-0.77008574],
       [-0.61491526],
       ...,
       [-0.44617371],
       [-0.58967362],
       [-0.12881968]])
```

2.2.2 encoding numerical features: teacher_number_of_previously_posted_projects

In [67]:

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

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

prev_projects_scalar = StandardScaler()
prev_projects_scalar.fit((X_train['teacher_number_of_previously_posted_projects'].values.as
print(f"Mean : {prev_projects_scalar.mean_[0]}, Standard deviation : {np.sqrt(prev_projects

# Now standardize the data with above maen and variance.
prev_projects_standardized_train = prev_projects_scalar.transform((X_train['teacher_number_
prev_projects_standardized_cv = prev_projects_scalar.transform((X_cv['teacher_number_of_pre
prev_projects_standardized_test = prev_projects_scalar.transform((X_test['teacher_number_of
print(prev_projects_standardized_train.shape,y_train.shape)
print(prev_projects_standardized_cv.shape,y_cv.shape)
print(prev_projects_standardized_test.shape,y_test.shape)
```

```
Mean : 11.66473602138561, Standard deviation : 29.394684894505467
(22445, 1) (22445, 1)
(11055, 1) (11055, 1)
(16500, 1) (16500, 1)
```

In [68]:

```
print(prev_projects_standardized_train)
```

```
[[ 0.75984022]
 [-0.22673269]
 [-0.32879196]
 ...
 [-0.36281171]
 [-0.32879196]
 [-0.32879196]]
```

2.2.3 encoding numerical features: quantity

In [69]:

```
quantity_scalar = StandardScaler()
quantity_scalar.fit((X_train['quantity'].values.astype(float)).reshape(-1,1)) # finding the
print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation : {np.sqrt(quantity_scalar.va

# Now standardize the data with above maen and variance.
quantity_standardized_train = quantity_scalar.transform((X_train['quantity'].values.astype(
quantity_standardized_cv = quantity_scalar.transform((X_cv['quantity'].values.astype(float)
quantity_standardized_test = quantity_scalar.transform((X_test['quantity'].values.astype(fl
print(quantity_standardized_train.shape,y_train.shape)
print(quantity_standardized_cv.shape,y_cv.shape)
print(quantity_standardized_test.shape,y_test.shape)
```

```
Mean : 17.001470260637113, Standard deviation : 26.03701886058425
(22445, 1) (22445, 1)
(11055, 1) (11055, 1)
(16500, 1) (16500, 1)
```

In [70]:

```
print(quantity_standardized_train)
```

```
[[-0.57615929]
 [ 0.1151641 ]
 [-0.26890445]
 ...
 [-0.49934558]
 [-0.19209074]
 [-0.26890445]]
```

2.3 Vectorizing Text data

2.3.1. Bag of words on preprocessed_essays

In [71]:

```
# We are considering only the words which appeared in at least 10 documents(rows or project
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)


vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2), max_features = 5000)
vectorizer.fit(X_train['preprocessed_essays'].values) # fit has to happen only on train dat

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer.transform(X_train['preprocessed_essays'].values)
X_cv_essay_bow = vectorizer.transform(X_cv['preprocessed_essays'].values)
X_test_essay_bow = vectorizer.transform(X_test['preprocessed_essays'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```

```
(22445, 24) (22445, 1)
(11055, 23) (11055, 1)
(16500, 23) (16500, 1)
============================================================================
========================
After vectorizations
(22445, 5000) (22445, 1)
(11055, 5000) (11055, 1)
(16500, 5000) (16500, 1)
============================================================================
========================
```

2.3.2 Bag of words on preprocessed_titles

In [73]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)


vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2),max_features = 5000)
vectorizer.fit(X_train['preprocessed_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_bow = vectorizer.transform(X_train['preprocessed_title'].values)
X_cv_title_bow = vectorizer.transform(X_cv['preprocessed_title'].values)
X_test_title_bow = vectorizer.transform(X_test['preprocessed_title'].values)

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
print("="*100)
```

```
(22445, 24) (22445, 1)
(11055, 23) (11055, 1)
(16500, 23) (16500, 1)
============================================================================
========================
After vectorizations
(22445, 1885) (22445, 1)
(11055, 1885) (11055, 1)
(16500, 1885) (16500, 1)
============================================================================
========================
```

2.3.3 TFIDF vectorizer on preprocessed_essays

In [74]:

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

print("="*100)
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features = 5000)
vectorizer.fit(X_train['preprocessed_essays'].values) # fit has to happen only on train dat

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer.transform(X_train['preprocessed_essays'].values)
X_cv_essay_tfidf = vectorizer.transform(X_cv['preprocessed_essays'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['preprocessed_essays'].values)

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

```
(22445, 24) (22445, 1)
(11055, 23) (11055, 1)
(16500, 23) (16500, 1)
============================================================================
========================
After vectorizations
(22445, 5000) (22445, 1)
(11055, 5000) (11055, 1)
(16500, 5000) (16500, 1)
============================================================================
========================
```

2.3.4 TFIDF vectorization on preprocessed_titles

In [75]:

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

print("="*100)
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=5,ngram_range=(1,2),max_features = 5000)
vectorizer.fit(X_train['preprocessed_title'].values) # fit has to happen only on train datc

# we use the fitted CountVectorizer to convert the text to vector
X_train_titles_tfidf = vectorizer.transform(X_train['preprocessed_title'].values)
X_cv_titles_tfidf = vectorizer.transform(X_cv['preprocessed_title'].values)
X_test_titles_tfidf = vectorizer.transform(X_test['preprocessed_title'].values)

print("After vectorizations")
print(X_train_titles_tfidf.shape, y_train.shape)
print(X_cv_titles_tfidf.shape, y_cv.shape)
print(X_test_titles_tfidf.shape, y_test.shape)
print("="*100)
```

```
(22445, 24) (22445, 1)
(11055, 23) (11055, 1)
(16500, 23) (16500, 1)
============================================================================
========================
After vectorizations
(22445, 3844) (22445, 1)
(11055, 3844) (11055, 1)
(16500, 3844) (16500, 1)
============================================================================
========================
```

2.3.5. Using Pretrained Models: Avg W2V

In [76]:

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

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

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

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

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

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

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

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


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickl

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


'''
```

Out[76]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
```

```
9/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGlo
veModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(glove
File,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n
splitLine = line.split()\n        word = splitLine[0]\n        embedding =
np.array([float(val) for val in splitLine[1:]])\n        model[word] = emb
edding\n    print ("Done.",len(model)," words loaded!")\n    return model
\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ====================
========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/
s]\nDone. 1917495  words loaded!\n\n# ============================\n\nword
s = []\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor
i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the w
ords in the coupus", len(words))\nwords = set(words)\nprint("the unique wo
rds in the coupus", len(words))\n\ninter_words = set(model.keys()).interse
ction(words)\nprint("The number of words that are present in both glove ve
ctors and our coupus",       len(inter_words),"(",np.round(len(inter_word
s)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.
keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus
[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n# stro
nging variables into pickle files python: http://www.jessicayung.com/how-t
o-use-pickle-to-save-and-load-variables-in-python/\n\nimport (http://www.j
essicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\n
import) pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dum
p(words_courpus, f)\n\n\n'
```

In [77]:

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

2.3.5.1 Using Pretrained Models: Avg W2V on preprocessed_essays

In [78]:

```python
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essays_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_essays'].values, position= 0, leave= True): # fo
    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_essays_train.append(vector)

print(len(avg_w2v_essays_train))
print(len(avg_w2v_essays_train[0]))
```

```
100%|██████████████████████████████████| 22445/22445 [00:15<00:00, 1411.21i
t/s]

22445
300
```

In [79]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essays_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_essays'].values, position= 0, leave= True): # for e
    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_essays_cv.append(vector)

print(len(avg_w2v_essays_cv))
print(len(avg_w2v_essays_cv[0]))
```

```
100%|██████████████████████████████████| 11055/11055 [00:07<00:00, 1427.45i
t/s]


11055
300
```

In [80]:

```
avg_w2v_essays_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays'].values, position= 0, leave= True): # for
    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_essays_test.append(vector)
print(len(avg_w2v_essays_test))
print(len(avg_w2v_essays_test[0]))
```

```
100%|██████████████████████████████████| 16500/16500 [00:11<00:00, 1466.38i
t/s]

16500
300
```

2.3.5.2 Using Pretrained Models: Avg W2V on project_title

In [81]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_titles_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_title'].values, position= 0, leave= True): # for
    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_titles_train.append(vector)

print(len(avg_w2v_titles_train))
print(len(avg_w2v_titles_train[0]))
#print(avg_w2v_titles_train[0])
```

```
100%|█████████████████| 22445/22445 [00:00<00:00, 24979.05i
t/s]

22445
300
```

In [82]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_titles_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_title'].values, position= 0, leave= True): # for ea
    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_titles_cv.append(vector)

print(len(avg_w2v_titles_cv))
print(len(avg_w2v_titles_cv[0]))
#print(avg_w2v_titles_train[0])
```

```
100%|█████████████████| 11055/11055 [00:00<00:00, 27449.00i
t/s]

11055
300
```

In [83]:

```
avg_w2v_titles_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_title'].values, position= 0, leave= True): # for
    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_titles_test.append(vector)
```

```
100%|██████████████████████████████| 16500/16500 [00:00<00:00, 27742.61i
t/s]
```

2.3.5.3 Using Pretrained Models: TFIDF weighted W2V on preprocessed_essays

In [84]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_essays'].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 [85]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_essay_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_essays'].values, position= 0, leave= True): # fo
    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 words
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_essay_train.append(vector)

print(len(tfidf_w2v_essay_train))
print(len(tfidf_w2v_essay_train[0]))
```

```
100%|████████████████████████████████| 22445/22445 [02:00<00:00, 185.81i
t/s]

22445
300
```

In [86]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_essay_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_essays'].values, position= 0, leave= True): # for e
    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 words
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_essay_cv.append(vector)

print(len(tfidf_w2v_essay_cv))
print(len(tfidf_w2v_essay_cv[0]))
```

```
100%|██████████████████████████████████████| 11055/11055 [00:59<00:00, 184.52i
t/s]

11055
300
```

In [87]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_essay_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays'].values, position= 0, leave= True): # for
    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 words
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_essay_test.append(vector)

print(len(tfidf_w2v_essay_test))
print(len(tfidf_w2v_essay_test[0]))
```

```
100%|██████████████████████████████████████| 16500/16500 [01:30<00:00, 183.29i
t/s]

16500
300
```

2.3.5.4 Using Pretrained Models: TFIDF weighted W2V on Project_title

In [88]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_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 [89]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_title_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_title'].values, position= 0, leave= True): # for
    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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title_train.append(vector)

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

```
100%|██████████| 22445/22445 [00:01<00:00, 11738.99i
t/s]

22445
300
```

In [90]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_title_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_title'].values, position= 0, leave= True): # for ea
    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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title_cv.append(vector)

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

```
100%|█████████████████████████████████| 11055/11055 [00:00<00:00, 12743.14i
t/s]

11055
300
```

In [91]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_title_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_title'].values, position= 0, leave= True): # for
    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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_title_test.append(vector)

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

```
100%|█████████████████████████████████| 16500/16500 [00:01<00:00, 13907.18i
t/s]

16500
300
```

2.3.6 Merging all the above features

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

SET 1

In [94]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_1 = hstack((X_train_essay_bow, X_train_title_bow, tr_clean_cat_encd , tr_clean_subcat_
X_cr_1 = hstack((X_cv_essay_bow, X_cv_title_bow, cv_clean_cat_encd , cv_clean_subcat_encd ,
X_te_1 = hstack((X_test_essay_bow, X_test_title_bow, te_clean_cat_encd , te_clean_subcat_er

print("Final Data matrix")
print(X_tr_1.shape, y_train.shape)
print(X_cr_1.shape, y_cv.shape)
print(X_te_1.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 6898) (22445, 1)
(11055, 6898) (11055, 1)
(16500, 6898) (16500, 1)
============================================================================
========================
```

SET 2

In [95]:

```
# merge two sparse matrices: httpss://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_2 = hstack((X_train_essay_tfidf, X_train_titles_tfidf, tr_clean_cat_encd , tr_clean_su
X_cr_2 = hstack((X_cv_essay_tfidf, X_cv_titles_tfidf, cv_clean_cat_encd , cv_clean_subcat_e
X_te_2 = hstack((X_test_essay_tfidf, X_test_titles_tfidf, te_clean_cat_encd , te_clean_subc

print("Final Data matrix")
print(X_tr_2.shape, y_train.shape)
print(X_cr_2.shape, y_cv.shape)
print(X_te_2.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 8857) (22445, 1)
(11055, 8857) (11055, 1)
(16500, 8857) (16500, 1)
============================================================================
========================
```

SET 3

In [96]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_3 = hstack((avg_w2v_essays_train, avg_w2v_titles_train, tr_clean_cat_encd , tr_clean_s
X_cr_3 = hstack((avg_w2v_essays_cv, avg_w2v_titles_cv, cv_clean_cat_encd , cv_clean_subcat_
X_te_3 = hstack((avg_w2v_essays_test, avg_w2v_titles_test, te_clean_cat_encd , te_clean_sub


print("Final Data matrix")
print(X_tr_3.shape, y_train.shape)
print(X_cr_3.shape, y_cv.shape)
print(X_te_3.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 613) (22445, 1)
(11055, 613) (11055, 1)
(16500, 613) (16500, 1)
============================================================================
========================
```

SET 4

In [97]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_4 = hstack((tfidf_w2v_essay_train, tfidf_w2v_title_train, tr_clean_cat_encd , tr_clear
X_cr_4 = hstack((tfidf_w2v_essay_cv, tfidf_w2v_title_cv, cv_clean_cat_encd , cv_clean_subca
X_te_4 = hstack((tfidf_w2v_essay_test, tfidf_w2v_title_test, te_clean_cat_encd , te_clean_s

print("Final Data matrix")
print(X_tr_2.shape, y_train.shape)
print(X_cr_2.shape, y_cv.shape)
print(X_te_2.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 8857) (22445, 1)
(11055, 8857) (11055, 1)
(16500, 8857) (16500, 1)
============================================================================
========================
```

In [98]:

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

Assignment 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, 0.5]

1. **Apply both Random Forrest and GBDT on these feature sets**

   - Set 1: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
   - Set 2: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
   - Set 3: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V). Here for this set take **20K** datapoints only.
   - Set 4: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V). Here for this set take **20K** datapoints only.

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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
   - Find the best hyper paramter using simple cross validation data
   - 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 (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html) 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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points

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 (http://zetcode.com/python/prettytable/)](http://zetcode.com/python/prettytable/)

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. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)](https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

3. Random Forest and GBDT

3.1 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

3.1.1 Applying Random Forests on BOW, SET 1

3.1.1.1 Hyper parameter Tuning using simple for loop

In [159]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your tr_loop will be 49041 - 49041%1000 = 4900
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

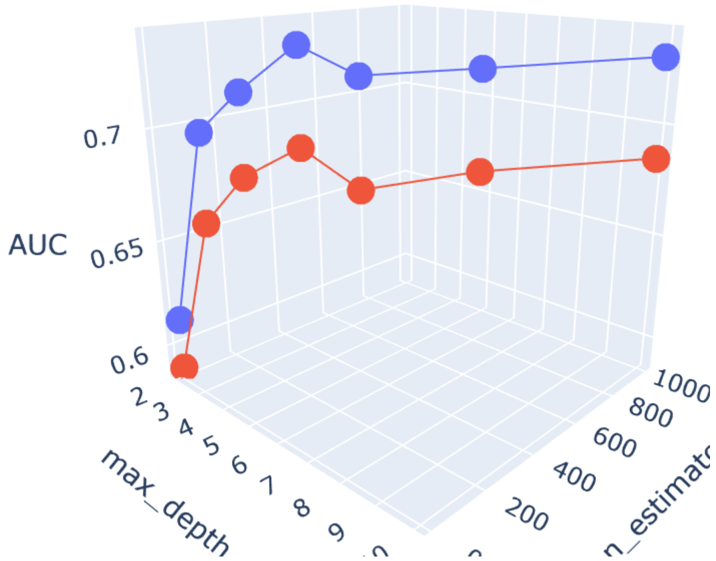
In [160]:

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

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

"""
def hyper_param_tune(X_tr, y_train, X_cr, y_cv):
    train_auc = []
    cv_auc = []
    depth = [2,3,4,5,6,8,10]
    n_estimators = [10,50,100,200,300,500,1000]
    for i in tqdm(depth, position= 0, leave= True):
        for j in tqdm(n_estimators, position= 0, leave= True):
            rf = RandomForestClassifier(max_depth=i,n_estimators= j, class_weight= "balance
            rf.fit(X_tr, y_train)

            y_train_pred = batch_predict(rf, X_tr)
            y_cv_pred = batch_predict(rf, X_cr)

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

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

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

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

In [112]:

```
hyper_param_tune(X_tr_1, np.ravel(y_train), X_cr_1, np.ravel(y_cv))
```

```
100%|██████████████████████████████████████████| 7/7 [01:36<00:00, 18.84s/
it]
100%|██████████████████████████████████████████| 7/7 [01:37<00:00, 19.14s/
it]
100%|██████████████████████████████████████████| 7/7 [01:40<00:00, 19.61s/
it]
100%|██████████████████████████████████████████| 7/7 [01:39<00:00, 19.82s/
it]
100%|██████████████████████████████████████████| 7/7 [01:42<00:00, 20.16s/
it]
100%|██████████████████████████████████████████| 7/7 [01:53<00:00, 23.40s/
it]
100%|██████████████████████████████████████████| 7/7 [02:07<00:00, 26.11s/
it]
100%|█████████████████████████████████████████| 7/7 [12:17<00:00, 110.92s/
it]
```



In [117]:

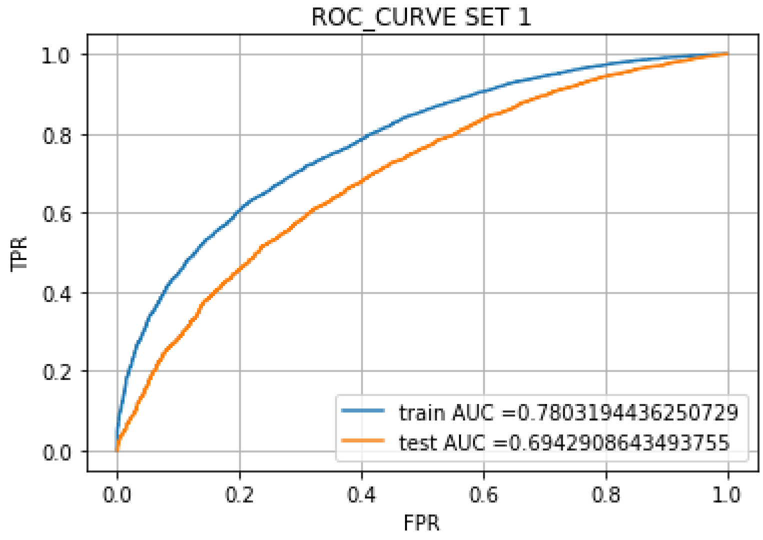
```
#flattening 2D to 1D array
y_train= np.ravel(y_train)
print(y_train.shape)
print("*"*20)
y_cv= np.ravel(y_cv)
print(y_cv.shape)
print("*"*20)
y_test= np.ravel(y_test)
print(y_test.shape)
print("*"*20)
```

```
(22445,)
********************
(11055,)
********************
(16500,)
********************
```

3.1.1.2 Plotting roc_curve

In [119]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.
from sklearn.metrics import roc_curve, auc
best_depth = 5
best_esti = 200
rf = RandomForestClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "
rf.fit(X_tr_1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(rf, X_tr_1)
y_test_pred = batch_predict(rf, X_te_1)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 1")
plt.grid()
plt.show()
```



3.1.1.3 Confusion Matrix

In [161]:

```
# 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
    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 [121]:

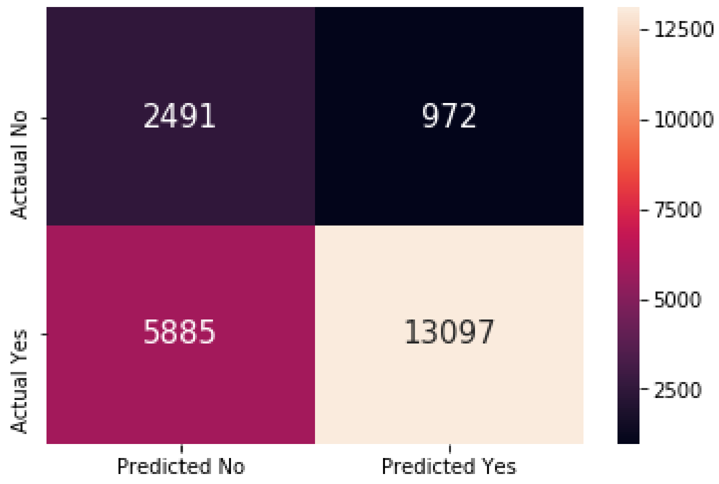
```
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)

def get_confusion_matrix(y_train, y_train_pred):
    cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
    sns.heatmap(cm, annot = True, fmt= 'd',annot_kws={"size": 15}, xticklabels= ['Predicted
```

```
the maximum value of tpr*(1-fpr) 0.4963077929079308 for threshold 0.5
```

In [122]:

```
print("Train Confusion Matrix")
get_confusion_matrix(y_train, y_train_pred)
```

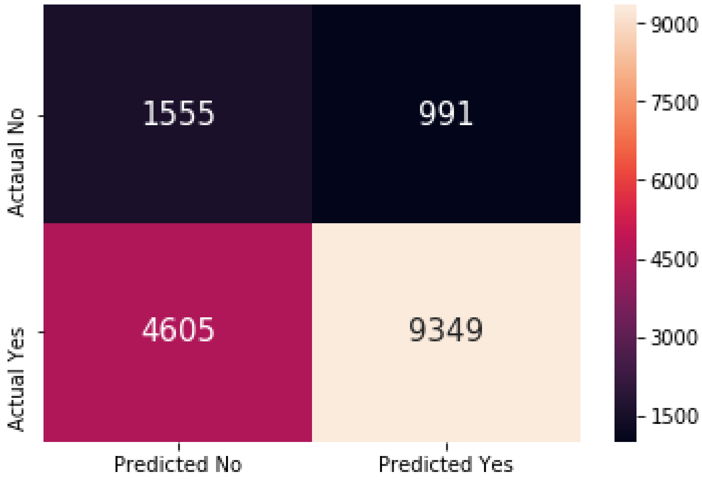
```
Train Confusion Matrix
```



In [124]:

```
print("Test Confusion Matrix")
get_confusion_matrix(y_test, y_test_pred)
```

```
Test Confusion Matrix
```



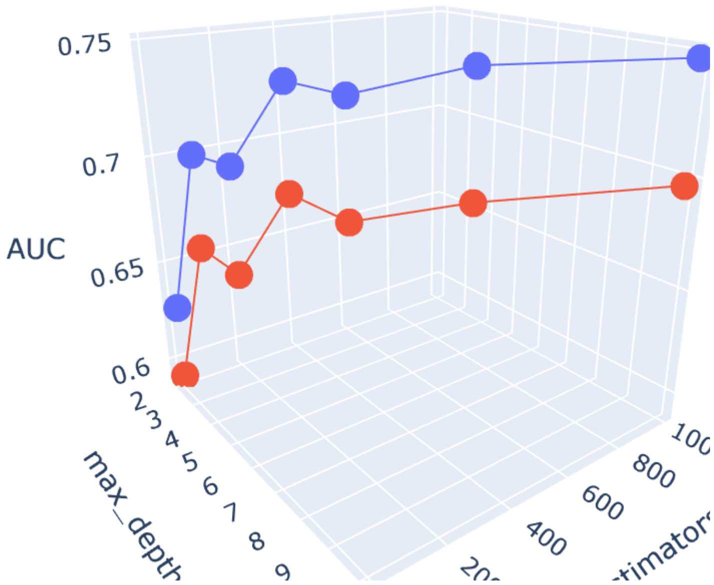
3.1.2 Applying Random Forests on TFIDF, SET 2

3.1.2.1 Hyper parameter Tuning using simple for loop

In [125]:

```
hyper_param_tune(X_tr_2, y_train, X_cr_2, y_cv)
```

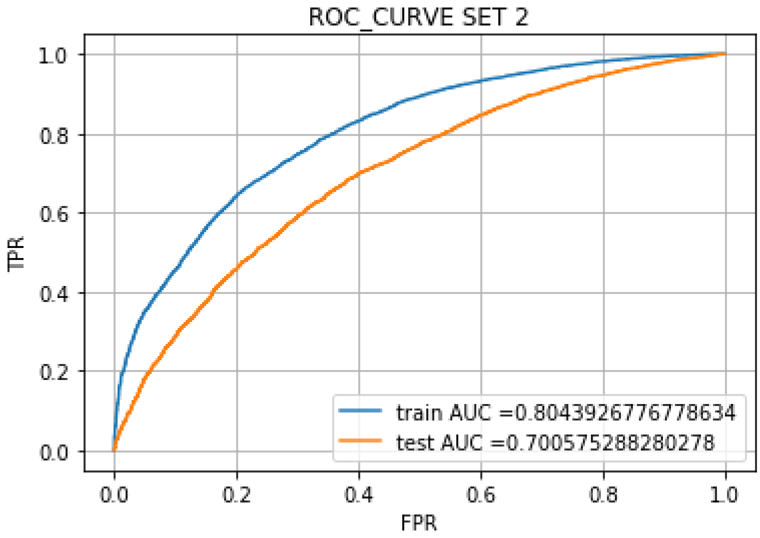
```
100%|██████████████████████████████████████████| 7/7 [01:25<00:00, 17.18s/
it]
100%|██████████████████████████████████████████| 7/7 [01:24<00:00, 16.79s/
it]
100%|██████████████████████████████████████████| 7/7 [01:31<00:00, 18.42s/
it]
100%|██████████████████████████████████████████| 7/7 [01:33<00:00, 19.08s/
it]
100%|██████████████████████████████████████████| 7/7 [01:45<00:00, 21.90s/
it]
100%|██████████████████████████████████████████| 7/7 [01:52<00:00, 23.24s/
it]
100%|██████████████████████████████████████████| 7/7 [02:17<00:00, 28.88s/
it]
100%|█████████████████████████████████████████| 7/7 [11:51<00:00, 111.19s/
it]
```



3.1.2.2 Plotting roc_curve

In [126]:

```
best_depth = 5
best_esti = 200
rf = RandomForestClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "
rf.fit(X_tr_2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(rf, X_tr_2)
y_test_pred = batch_predict(rf, X_te_2)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 2")
plt.grid()
plt.show()
```

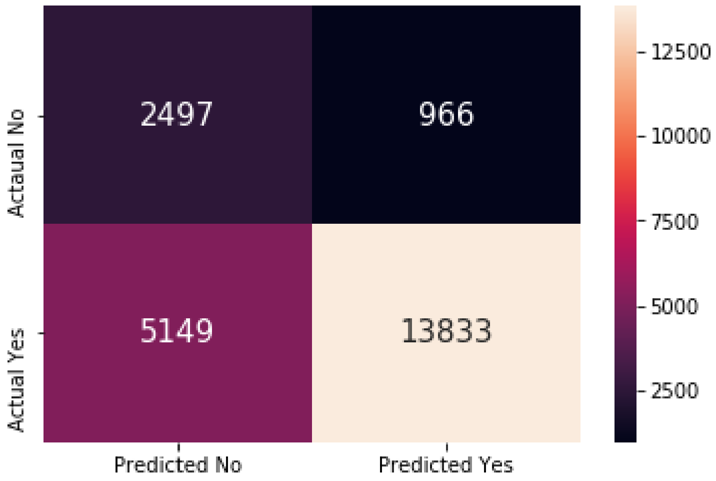


3.1.2.3 Confusion Matrix Set 2

In [127]:

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train Confusion Matrix")
get_confusion_matrix(y_train, y_train_pred)
```

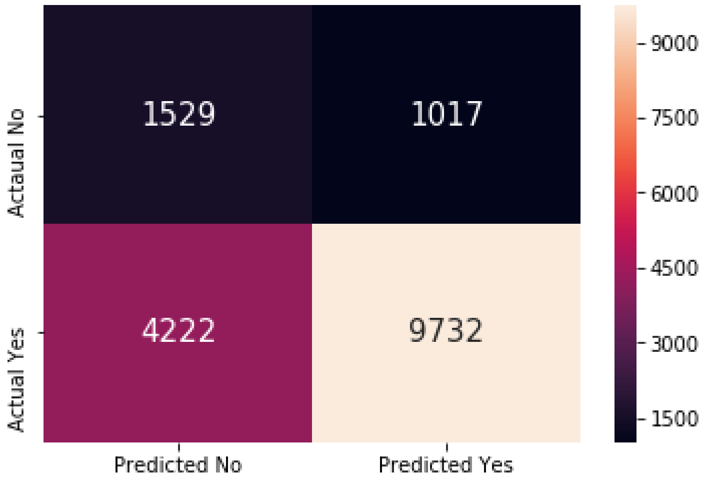
```
the maximum value of tpr*(1-fpr) 0.5254609645388629 for threshold 0.502
Train Confusion Matrix
```



In [128]:

```
print("Test Confusion Matrix")
get_confusion_matrix(y_test, y_test_pred)
```

```
Test Confusion Matrix
```



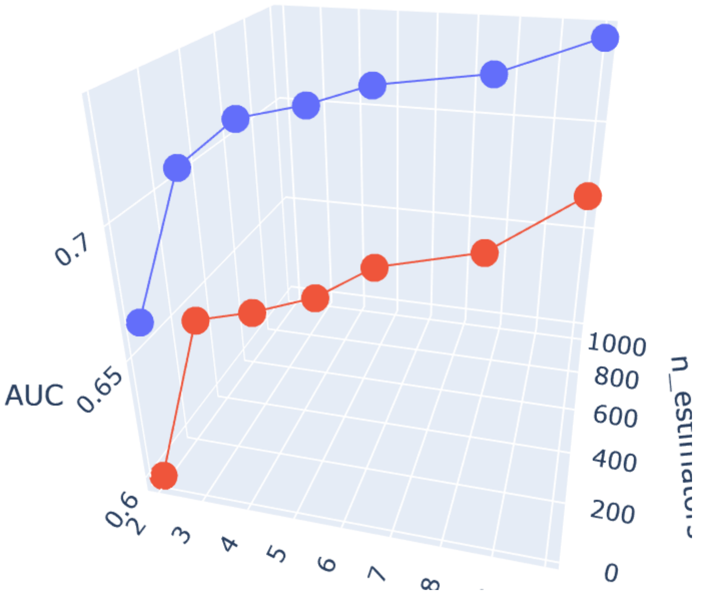
3.1.3 Applying Random Forests on AVG W2V, SET 3

3.1.3.1 Hyper parameter Tuning using simple for loop

In [129]:

```
hyper_param_tune((X_tr_3[0:10000]), (y_train[0:10000]), (X_cr_3[0:5000]), (y_cv[0:5000]))
```

```
100%|███████████████████████████████████████████| 7/7 [01:55<00:00, 24.84s/
it]
100%|███████████████████████████████████████████| 7/7 [02:32<00:00, 33.25s/
it]
100%|███████████████████████████████████████████| 7/7 [03:03<00:00, 39.23s/
it]
100%|███████████████████████████████████████████| 7/7 [03:47<00:00, 49.26s/
it]
100%|███████████████████████████████████████████| 7/7 [04:17<00:00, 56.22s/
it]
100%|███████████████████████████████████████████| 7/7 [06:44<00:00, 90.79s/
it]
100%|██████████████████████████████████████████| 7/7 [08:38<00:00, 112.52s/
it]
100%|██████████████████████████████████████████| 7/7 [30:58<00:00, 336.18s/
it]
```



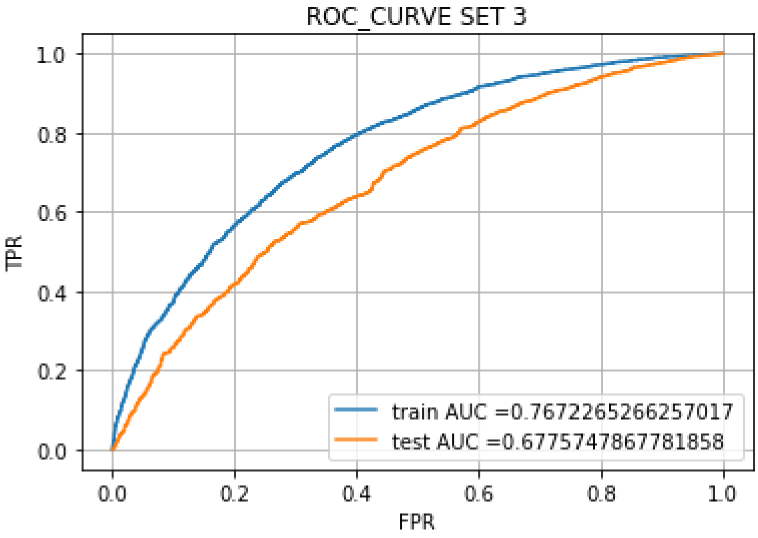
In [ ]:

```
#X_tr_3[0:10000].shape
#np.ravel(y_train)[0:10000].shape
```

3.1.3.2 Plotting roc_curve

In [132]:

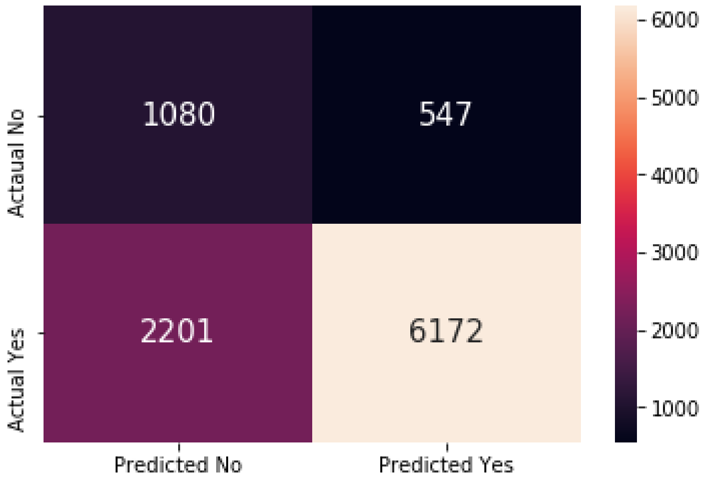
```
best_depth = 3
best_esti = 50
rf = RandomForestClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "
rf.fit((X_tr_3[0:10000]), (y_train[0:10000]))
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(rf, (X_tr_3[0:10000]))
y_test_pred = batch_predict(rf, (X_te_3[0:5000]))

train_fpr, train_tpr, tr_thresholds = roc_curve((y_train[0:10000]), y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test[0:5000], 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("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 3")
plt.grid()
plt.show()
```



In [133]:

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train Confusion Matrix")
get_confusion_matrix(y_train[0:10000], y_train_pred)
```

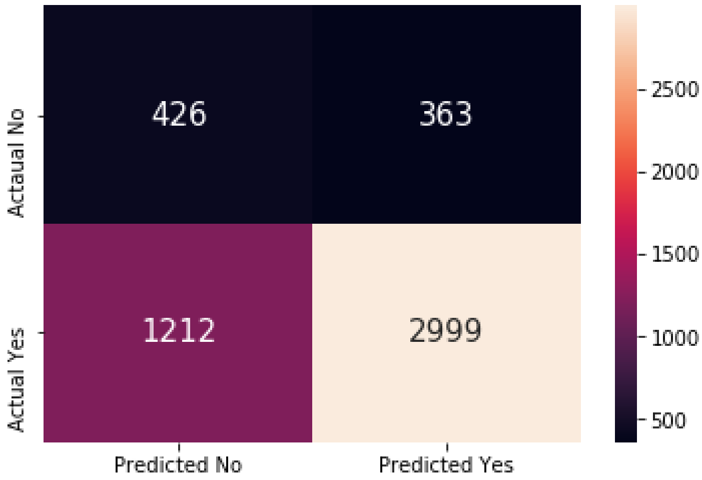
```
the maximum value of tpr*(1-fpr) 0.48930654925822903 for threshold 0.5
Train Confusion Matrix
```



In [134]:

```
print("Test Confusion Matrix")
get_confusion_matrix(y_test[0:5000], y_test_pred)
```

```
Test Confusion Matrix
```



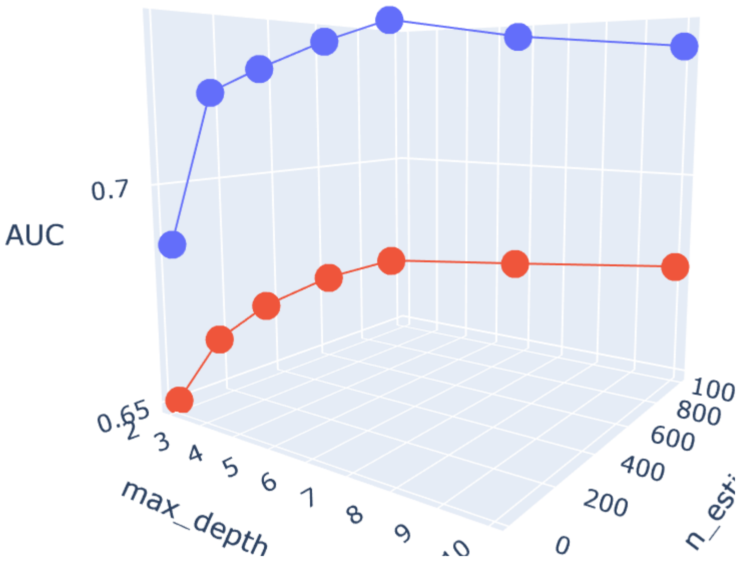
3.1.4 Applying Random Forests on TFIDF W2V, SET 4

3.1.4.1 Hyper parameter Tuning using simple for loop

In [136]:

```
hyper_param_tune((X_tr_4[0:10000]), (y_train[0:10000]), (X_cr_4[0:5000]),(y_cv[0:5000]))
```

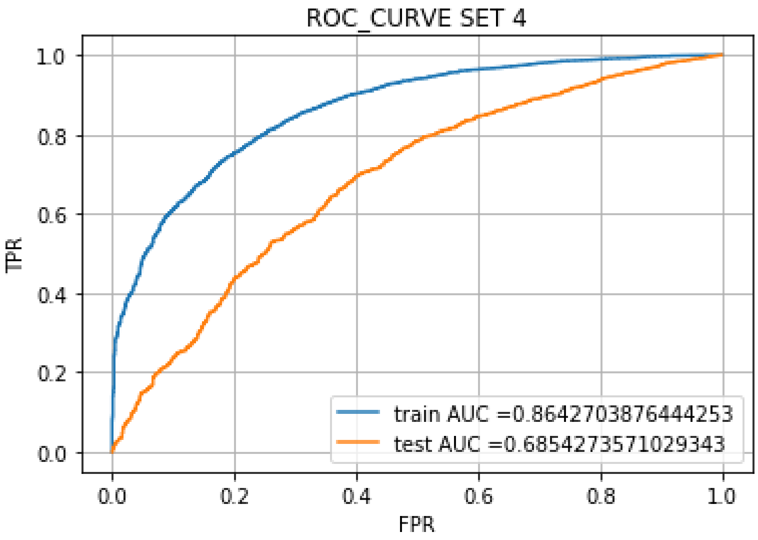
```
100%|████████████████████████████████████████| 7/7 [01:59<00:00, 25.96s/
it]
100%|████████████████████████████████████████| 7/7 [02:36<00:00, 33.56s/
it]
100%|████████████████████████████████████████| 7/7 [03:12<00:00, 41.60s/
it]
100%|████████████████████████████████████████| 7/7 [03:54<00:00, 51.37s/
it]
100%|████████████████████████████████████████| 7/7 [05:13<00:00, 68.89s/
it]
100%|████████████████████████████████████████| 7/7 [06:27<00:00, 83.70s/
it]
100%|███████████████████████████████████████| 7/7 [09:10<00:00, 123.92s/
it]
100%|███████████████████████████████████████| 7/7 [32:34<00:00, 352.56s/
it]
```



3.1.4.2 Plotting roc_curve

In [135]:

```
best_depth = 5
best_esti = 200
rf = RandomForestClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "
rf.fit(X_tr_4[0:10000], y_train[0:10000])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(rf, X_tr_4[0:10000])
y_test_pred = batch_predict(rf, X_te_4[0:5000])

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train[0:10000], y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test[0:5000], 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("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 4")
plt.grid()
plt.show()
```

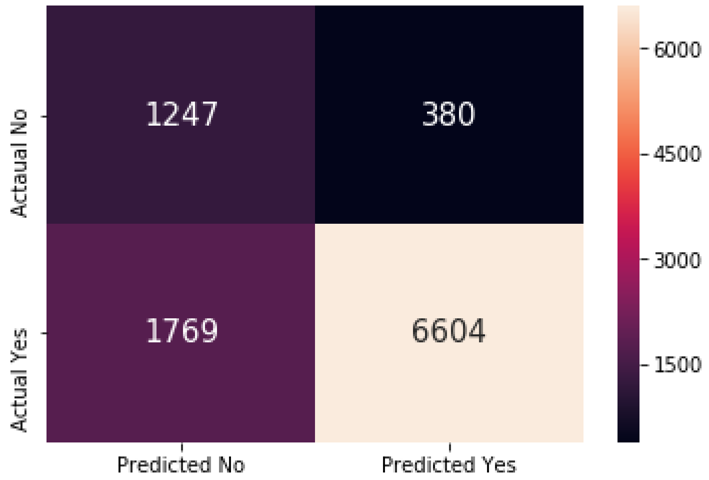


3.4.3 COnfusion Matrix

In [137]:

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train Confusion Matrix")
get_confusion_matrix(y_train[0:10000], y_train_pred)
```

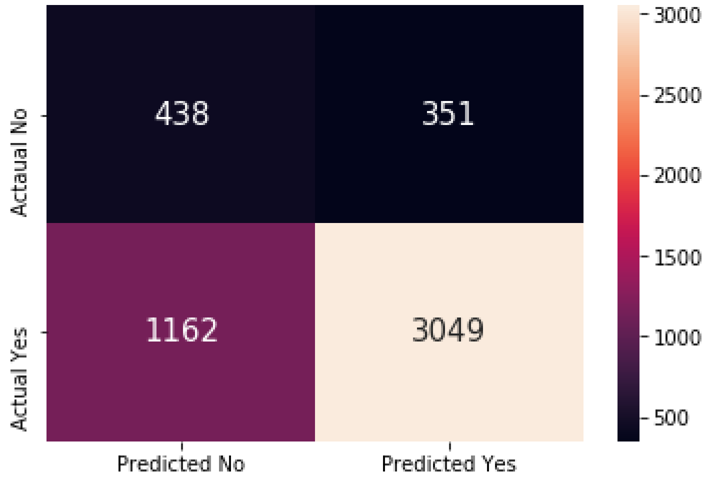
```
the maximum value of tpr*(1-fpr) 0.6045119270379936 for threshold 0.506
Train Confusion Matrix
```



In [138]:

```
print("Test Confusion Matrix")
get_confusion_matrix(y_test[0:5000], y_test_pred)
```

```
Test Confusion Matrix
```



3.2 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

3.2.1 Applying XGBOOST on BOW, SET 1

3.2.1.1 Hyperparameter tuning using simple for loop

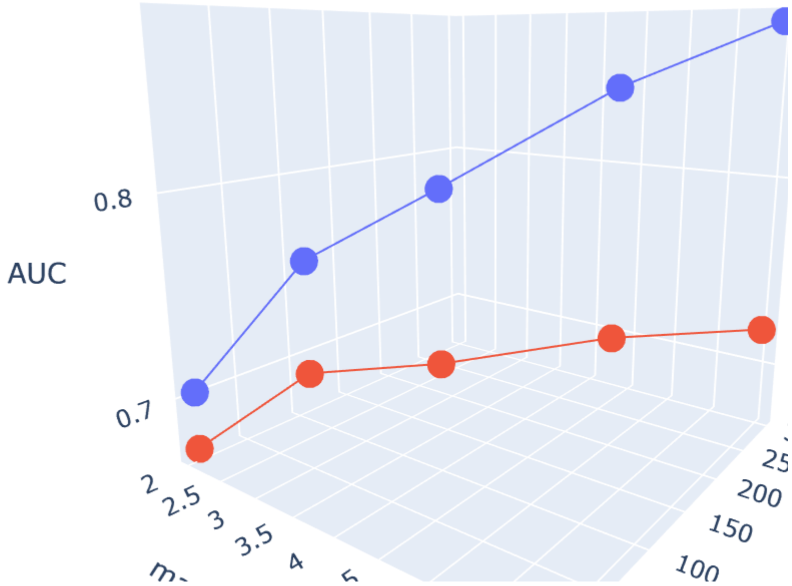
In [164]:

```
import xgboost as xgb
def hyper_param_tune2(X_tr, y_train, X_cr, y_cv):
    train_auc = []
    cv_auc = []
    depth = [2,3,4,5,6]
    n_estimators = [10,50,100,200,300]
    for i in tqdm(depth, position= 0, leave= True):
        for j in tqdm(n_estimators, position= 0, leave= True):
            gbdt = xgb.XGBClassifier(max_depth=i, n_estimators= j, class_weight= "balanced"
            gbdt.fit(X_tr, y_train)

            y_train_pred = batch_predict(gbdt, X_tr)
            y_cv_pred = batch_predict(gbdt, X_cr)

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

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

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

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

In [153]:

```
hyper_param_tune2(X_tr_1, y_train, X_cr_1, y_cv)
```

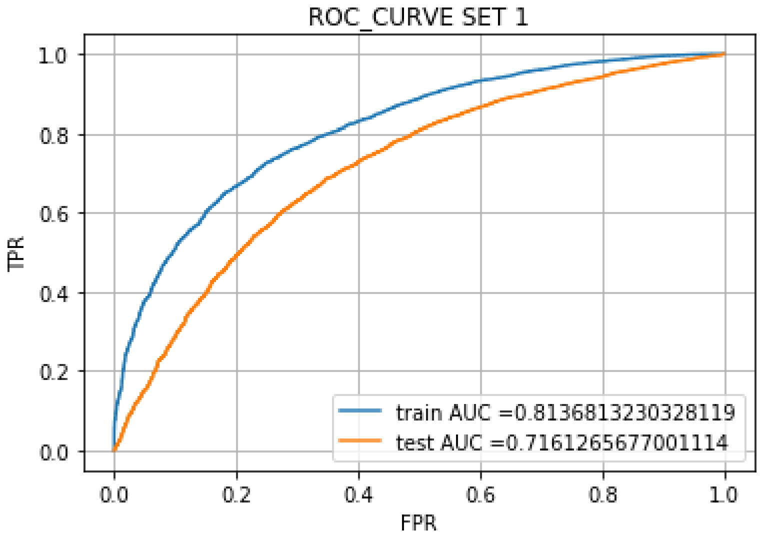
```
100%|██████████████████████████████████████████| 5/5 [03:30<00:00, 47.87s/
it]
100%|██████████████████████████████████████████| 5/5 [03:35<00:00, 49.34s/
it]
100%|██████████████████████████████████████████| 5/5 [04:20<00:00, 59.00s/
it]
100%|██████████████████████████████████████████| 5/5 [05:13<00:00, 71.70s/
it]
100%|██████████████████████████████████████████| 5/5 [05:29<00:00, 75.75s/
it]
100%|█████████████████████████████████████████| 5/5 [22:09<00:00, 275.63s/
it]
```



3.2.1.2 Plot RoC_corve

In [154]:

```
best_depth = 3
best_esti = 50
gbdt = xgb.XGBClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "bal
gbdt.fit(X_tr_1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(gbdt, X_tr_1)
y_test_pred = batch_predict(gbdt, X_te_1)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 1")
plt.grid()
plt.show()
```

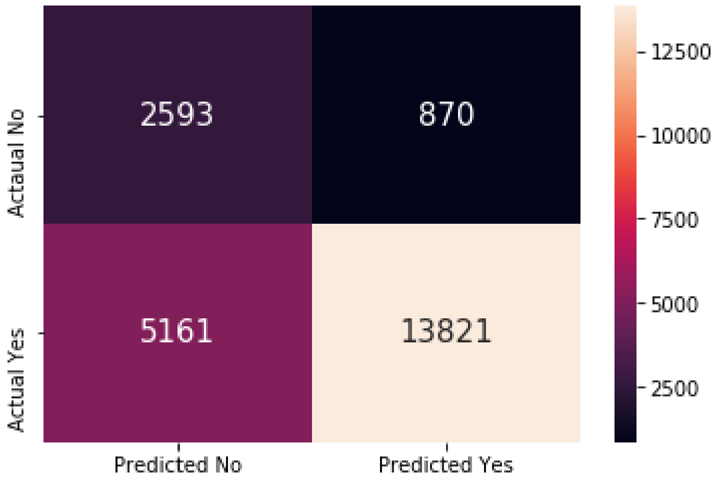


3.2.1.3 Confusion Matrix

In [155]:

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train Confusion Matrix")
get_confusion_matrix(y_train, y_train_pred)
```

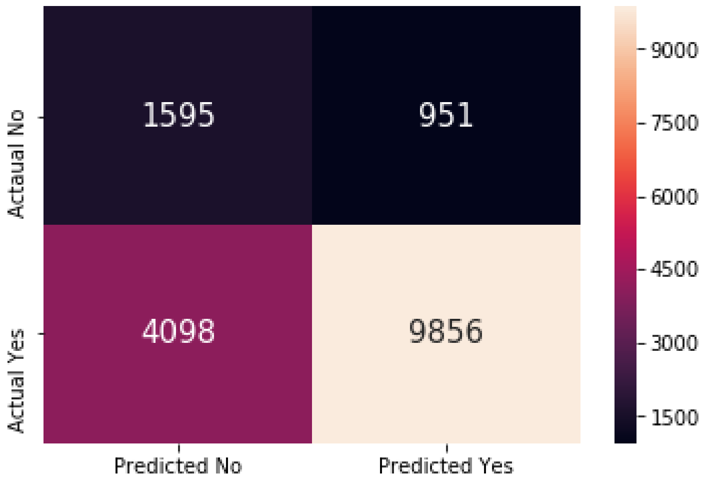
```
the maximum value of tpr*(1-fpr) 0.5451895503660124 for threshold 0.832
Train Confusion Matrix
```



In [156]:

```
print("Train Confusion Matrix")
get_confusion_matrix(y_test, y_test_pred)
```

```
Train Confusion Matrix
```



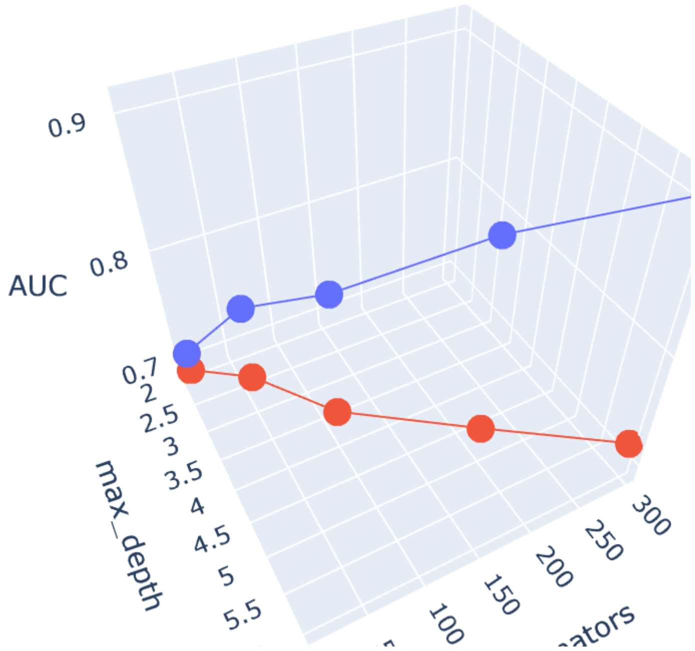
3.2.2 Applying XGBOOST on TFIDF, SET 2

3.2.2.1 Hyperparameter tuning using simple for loop

In [167]:

```
hyper_param_tune2(X_tr_2, y_train, X_cr_2, y_cv)
```

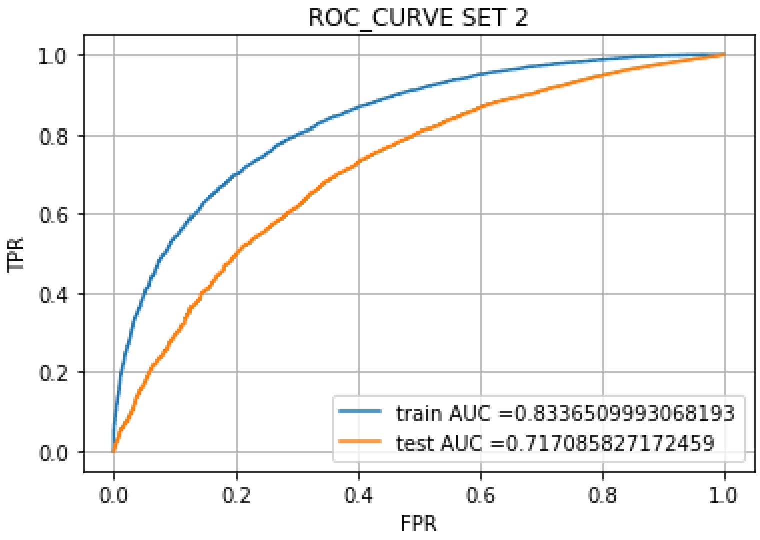
```
100%|███████████████████████████████████████████| 5/5 [05:18<00:00, 73.42s/
it]
100%|███████████████████████████████████████████| 5/5 [06:54<00:00, 95.90s/
it]
100%|██████████████████████████████████████████| 5/5 [08:51<00:00, 122.64s/
it]
100%|██████████████████████████████████████████| 5/5 [10:33<00:00, 146.57s/
it]
100%|██████████████████████████████████████████| 5/5 [12:52<00:00, 178.17s/
it]
100%|██████████████████████████████████████████| 5/5 [44:30<00:00, 562.11s/
it]
```



3.2.2.2 Plot ROC_curve

In [169]:

```
best_depth = 3
best_esti = 50
gbdt = xgb.XGBClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "bal
gbdt.fit(X_tr_2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(gbdt, X_tr_2)
y_test_pred = batch_predict(gbdt, X_te_2)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 2")
plt.grid()
plt.show()
```

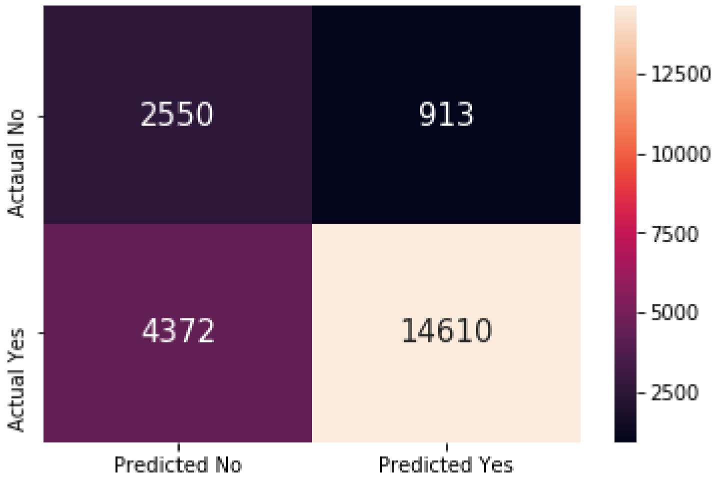


3.2.2.3 Confusion Matrix

In [170]:

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train Confusion Matrix")
get_confusion_matrix(y_train, y_train_pred)
```

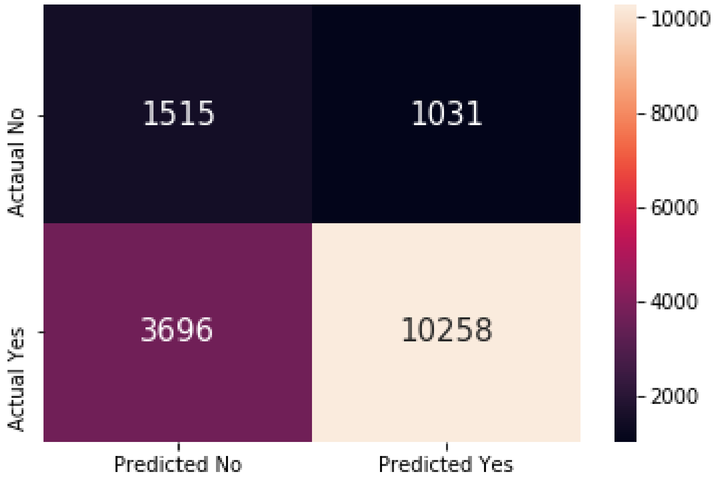
```
the maximum value of tpr*(1-fpr) 0.566755751067481 for threshold 0.822
Train Confusion Matrix
```



In [171]:

```
print("Train Confusion Matrix")
get_confusion_matrix(y_test, y_test_pred)
```

```
Train Confusion Matrix
```



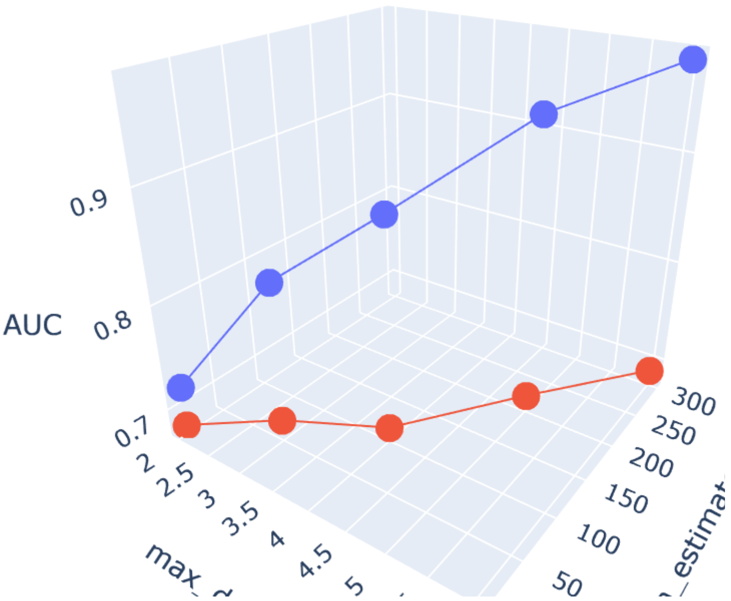
3.2.3 Applying XGBOOST on AVG W2V, SET 3

3.2.3.1 Hyper parameter tuning using simple for loop

In [172]:

```
hyper_param_tune2((X_tr_3[0:10000]), (y_train[0:10000]), (X_cr_3[0:5000]), (y_cv[0:5000]))
```

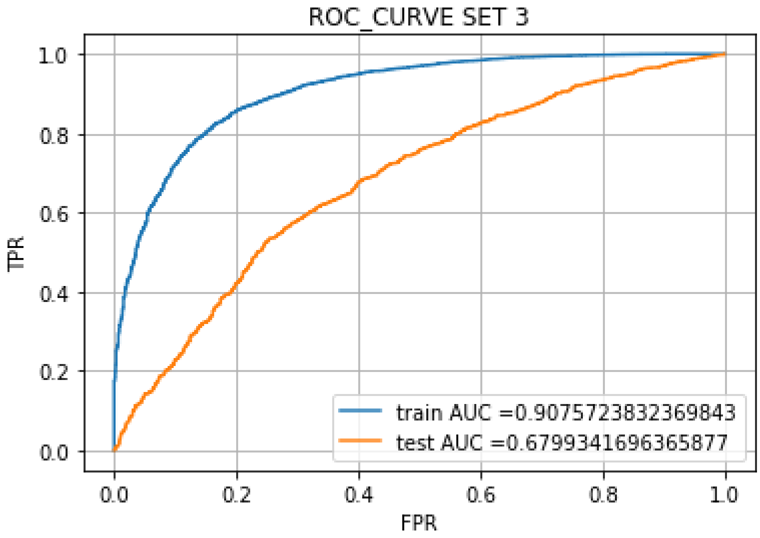
```
100%|██████████████████████████████████████████| 5/5 [07:00<00:00, 97.06s/
it]
100%|█████████████████████████████████████████| 5/5 [10:04<00:00, 139.69s/
it]
100%|█████████████████████████████████████████| 5/5 [12:15<00:00, 167.82s/
it]
100%|█████████████████████████████████████████| 5/5 [15:12<00:00, 212.66s/
it]
100%|█████████████████████████████████████████| 5/5 [16:27<00:00, 226.16s/
it]
100%|███████████████████████████████████████| 5/5 [1:01:00<00:00, 759.09s/
it]
```



3.2.3.1 Plot ROC_curve

In [176]:

```
best_depth = 3
best_esti = 50
gbdt = xgb.XGBClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "bal
gbdt.fit(X_tr_3[0:10000], y_train[0:10000])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(gbdt, X_tr_3[0:10000])
y_test_pred = batch_predict(gbdt, X_te_3[0:5000])

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train[0:10000], y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test[0:5000], 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("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 3")
plt.grid()
plt.show()
```

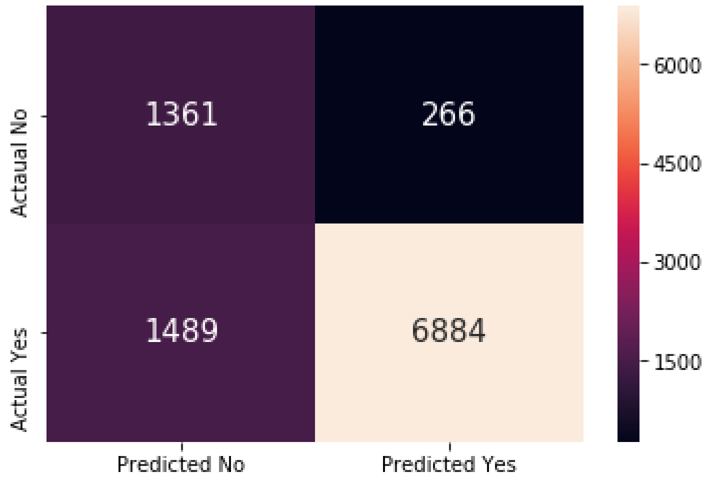


3.2.3.1 Confusion Matrix

In [177]:

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train Confusion Matrix")
get_confusion_matrix(y_train[0:10000], y_train_pred)
```

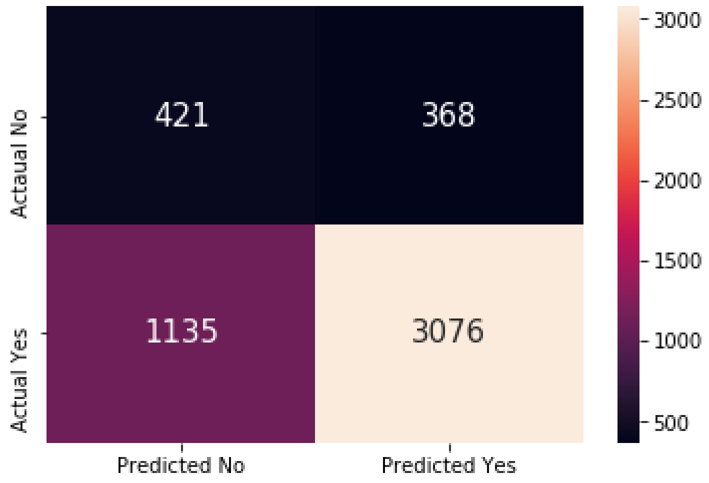
```
the maximum value of tpr*(1-fpr) 0.6877495940466587 for threshold 0.812
Train Confusion Matrix
```



In [178]:

```
print("Train Confusion Matrix")
get_confusion_matrix(y_test[0:5000], y_test_pred)
```

```
Train Confusion Matrix
```



In [ ]:

```

```

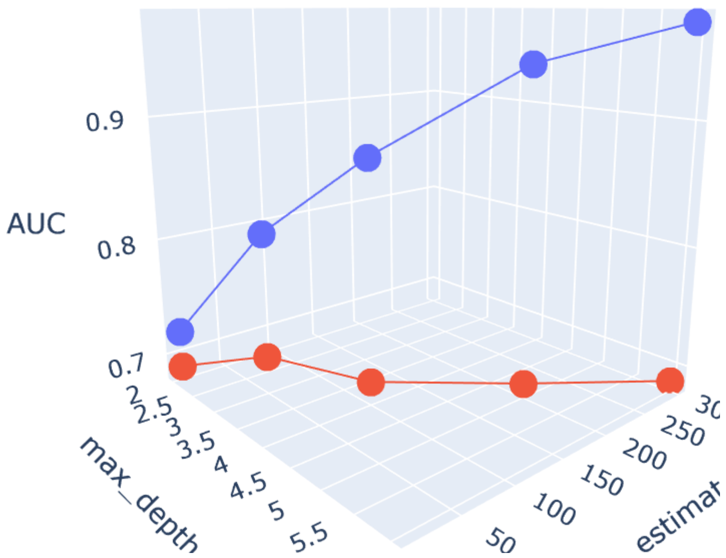
3.2.4 Applying XGBOOST on TFIDF W2V, SET 4

3.2.4.1 Hyper parameter tuning using simple for loop

In [179]:

```
hyper_param_tune2((X_tr_4[0:10000]), (y_train[0:10000]), (X_cr_4[0:5000]), (y_cv[0:5000]))
```

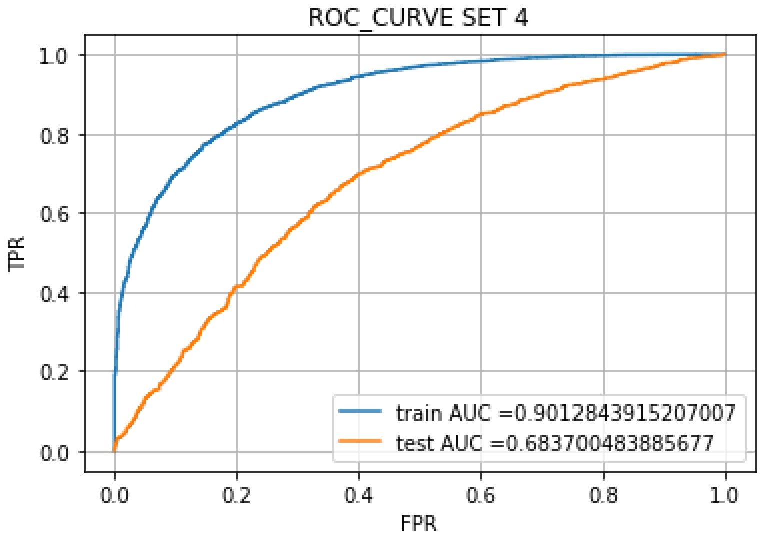
```
100%|██████████████████████████████████████████| 5/5 [07:05<00:00, 96.98s/
it]
100%|█████████████████████████████████████████| 5/5 [10:26<00:00, 144.08s/
it]
100%|█████████████████████████████████████████| 5/5 [13:23<00:00, 184.60s/
it]
100%|█████████████████████████████████████████| 5/5 [16:05<00:00, 224.23s/
it]
100%|█████████████████████████████████████████| 5/5 [17:23<00:00, 239.71s/
it]
100%|███████████████████████████████████████| 5/5 [1:04:24<00:00, 800.51s/
it]
```



3.2.4.2 Plot ROC_curve

In [187]:

```
best_depth = 3
best_esti = 50
gbdt = xgb.XGBClassifier(max_depth= best_depth, n_estimators= best_esti, class_weight= "bal
gbdt.fit(X_tr_4[0:10000], y_train[0:10000])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(gbdt, X_tr_4[0:10000])
y_test_pred = batch_predict(gbdt, X_te_4[0:5000])

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train[0:10000], y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test[0:5000], 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("FPR")
plt.ylabel("TPR")
plt.title("ROC_CURVE SET 4")
plt.grid()
plt.show()
```

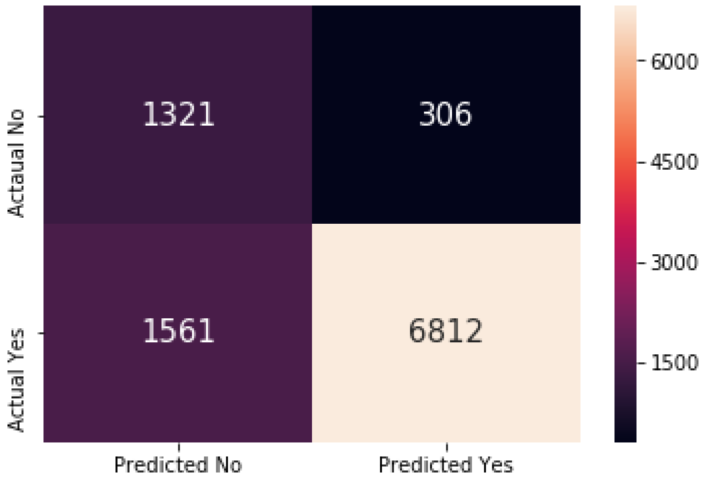


3.2.4.3 Confusion Matrix

In [188]:

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train Confusion Matrix")
get_confusion_matrix(y_train[0:10000], y_train_pred)
```

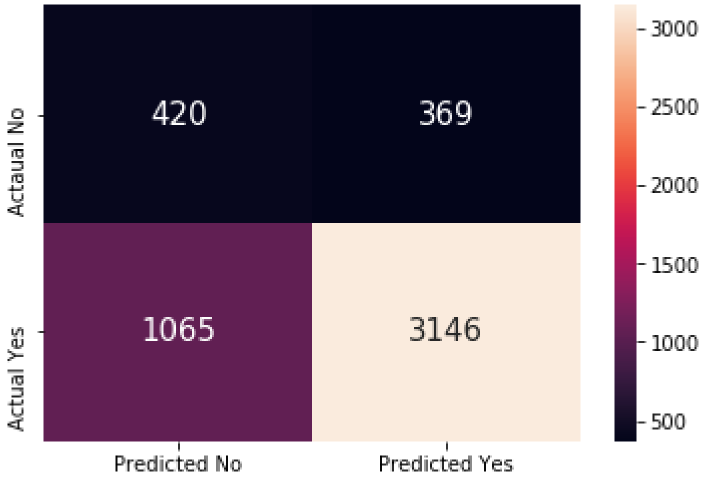
```
the maximum value of tpr*(1-fpr) 0.6605547391588747 for threshold 0.805
Train Confusion Matrix
```



In [189]:

```
print("Train Confusion Matrix")
get_confusion_matrix(y_test[0:5000], y_test_pred)
```

```
Train Confusion Matrix
```



3. Conclusion

In [190]:

```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer","Model","Max_Depth","No. of Esimators", "Train AUC", "Test Au
x.add_row(["BOW","Random Forest", 5, 200, 0.78, 0.69])
x.add_row(["-------------------","--------------------","----------------","-----------","---
x.add_row(["BOW","XGBoost", 3, 50, 0.81, 0.71])
x.add_row(["-------------------","--------------------","----------------","-----------","---
x.add_row(["TFIDF","Random Forest", 5, 200, 0.80, 0.70 ])
x.add_row(["-------------------","--------------------","----------------","-----------","---
x.add_row(["TFIDF","XGBoost", 3, 50, 0.83, 0.71 ])
x.add_row(["-------------------","--------------------","----------------","-----------","---
x.add_row(["AVG_W2V","Random Forest", 3, 50, 0.76, 0.67])
x.add_row(["-------------------","--------------------","----------------","-----------","---
x.add_row(["AVG_W2V","XGBoost", 3, 50, 0.90, 0.67])
x.add_row(["-------------------","--------------------","----------------","-----------","---
x.add_row(["TFIDF_W2V","Random Forest", 5, 200, 0.80, 0.68])
x.add_row(["-------------------","--------------------","----------------","-----------","---
x.add_row(["TFIDF_W2V","XGBoost", 3, 50, 0.90, 0.68])

print(x)
```

```
+---------------------+----------------------+-----------------+-----------
-------+-------------+-------------+
|     Vectorizer      |        Model         |    Max_Depth    | No. of Esi
mators |  Train AUC  |   Test Auc  |
+---------------------+----------------------+-----------------+-----------
-------+-------------+-------------+
|        BOW          |    Random Forest     |        5        |        200
|     0.78    |     0.69    |
| ------------------- | -------------------- | --------------- |   --------
---     | ----------- | ----------- |
|        BOW          |       XGBoost        |        3        |         50
|     0.81    |     0.71    |
| ------------------- | -------------------- | --------------- |   --------
---     | ----------- | ----------- |
|       TFIDF         |    Random Forest     |        5        |        200
|     0.8     |     0.7     |
| ------------------- | -------------------- | --------------- |   --------
---     | ----------- | ----------- |
|       TFIDF         |       XGBoost        |        3        |         50
|     0.83    |     0.71    |
| ------------------- | -------------------- | --------------- |   --------
---     | ----------- | ----------- |
|      AVG_W2V        |    Random Forest     |        3        |         50
|     0.76    |     0.67    |
| ------------------- | -------------------- | --------------- |   --------
---     | ----------- | ----------- |
|      AVG_W2V        |       XGBoost        |        3        |         50
|     0.9     |     0.67    |
| ------------------- | -------------------- | --------------- |   --------
---     | ----------- | ----------- |
|     TFIDF_W2V       |    Random Forest     |        5        |        200
|     0.8     |     0.68    |
| ------------------- | -------------------- | --------------- |   --------
---     | ----------- | ----------- |
|     TFIDF_W2V       |       XGBoost        |        3        |         50
|     0.9     |     0.68    |
+--------------------+----------------------+-----------------+-----------
-------+-------------+-------------+
```