DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

```
C:\Users\Kalyan\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, o
pening SSH/SCP/SFTP paths will be disabled.  `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install
paramiko` to suppress')
C:\Users\Kalyan\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; a
liasing 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' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

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

Out[4]:

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

1.2 preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

1.3 Text preprocessing

In [7]:

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

In [8]:

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

Out[8]:

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

In [9]:

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

In [10]:

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

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
```

```
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade.  This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of
my own pocket on resources to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students receive free or reduced pr
ice lunch.  Despite their disabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able
to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids don't want to sit and do worksheets. They want to learn to count by ju
mping and playing. Physical engagement is the key to our success. The number toss and color and sh
ape mats can make that happen. My students will forget they are doing work and just have the fun a
6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The grea
t teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% Af
rican-American, making up the largest segment of the student body. A typical school in Dallas is m
ade up of 23.2% African-American students. Most of the students are on free or reduced lunch. We a
ren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As
```

```
an educator I am inspiring minds of young children and we focus not only on academics but one smar
t, effective, efficient, and disciplined students with good character.In our classroom we can util
ize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the so
und enough to receive the message. Due to the volume of my speaker my students can't hear videos o
r books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my s
tudents will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will all
ow me to have more room for storage of things that are needed for the day and has an extra part to
it I can use.  The table top chart has all of the letter, words and pictures for students to learn
about different letters and it is more accessible.nannan
==================================================
```

In [11]:

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

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

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

In [12]:

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

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

In [13]:

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

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

In [14]:

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

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

In [15]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "do
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [16]:

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

```
100%|████████████████████████████████| 109248/109248 [01:42<00:00, 1063.37it/s]
```

In [17]:

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

Out[17]:

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

1.4 Preprocessing of `project_title`

In [18]:

```
print(project_data['project_title'].values[0])
```

```
Educational Support for English Learners at Home
```

In [19]:

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

```
We Need To Move It While We Input It!
==================================================
```

In [20]:

```
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
We Need To Move It While We Input It!
```

In [21]:

```
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
We Need To Move It While We Input It
```

In [22]:

```
from tqdm import tqdm
preprocessed_titles = []
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████████████████████████| 109248/109248 [00:03<00:00, 28600.95it/s]
```

In [23]:

```
preprocessed_essays[20000]
```

Out[23]:

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

1.5 Preparing data for models

In [24]:

```
project_data.columns
```

Out[24]:

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

we are going to consider

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

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

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

1.5.3 Vectorizing Numerical features

In [25]:

```
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 [26]:

```
catogories_grade = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list_grade = []
for i in catogories_grade:
    temp = ""
    for j in i:
        j = j.replace('-','_')
        j = j.replace(' ','_')
        temp+=j.strip('')
        temp = temp.replace('&','_')
    cat_list_grade.append(temp.strip())
print(cat_list_grade[0:5])
project_data['clean_grades'] = cat_list_grade
project_data.drop(['project_grade_category'], axis=1, inplace=True)
```

```
['Grades_PreK_2', 'Grades_6_8', 'Grades_6_8', 'Grades_PreK_2', 'Grades_PreK_2']
```

In [27]:

```
project_data['teacher_prefix'].fillna('NoValue',inplace=True)
```

In [28]:

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

Out[28]:

```
(109248, 20)
```

In [29]:

```
project_data["preprocessed_essays"]=preprocessed_essays
project_data["preprocessed_titles"]=preprocessed_titles
```

In [30]:

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

Out[30]:

```
(109248, 22)
```

Computing Sentiment Scores

In [31]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk
nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students w
ith the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelli
gences i use a wide range\
of techniques to help all my students succeed students in my class come from a variety of differen
t backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a carin
g community of successful \
learners which can be seen through collaborative student project based learning in and out of the
classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities to practice
a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the ki
ndergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role pla
y in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food
i will take their idea \
and create common core cooking lessons where we learn important math and writing concepts while co
```

```
oking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that went into maki
ng the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project woul
d expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade apple
sauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create our own cook
books to be printed and \
shared with families students will gain math and literature skills as well as a life long enjoymen
t for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

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

The twython library has not been installed. Some functionality from the twitter package will not b
e available.
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Kalyan\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

Assignment 5: Logistic Regression

1. **[Task-1] Logistic Regression(either SGDClassifier with log loss, or LogisticRegression) on these feature sets**

    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (`BOW with bi-grams` with `min_df=10` and `max_features=5000`)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (`TFIDF with bi-grams` with `min_df=10` and `max_features=5000`)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

2. **Hyper paramter tuning (find best hyper parameters corresponding the algorithm that you choose)**

    - Find the best hyper parameter which will give the maximum AUC value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **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.
    - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
    - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

4. **[Task-2] Apply Logistic Regression on the below feature set Set 5 by finding the best hyper parameter as suggested in step 2 and step 3.**
5. Consider these set of features Set 5 :

    - **school_state** : categorical data
    - **clean_categories** : categorical data
    - **clean_subcategories** : categorical data
    - **project_grade_category** :categorical data
    - **teacher_prefix** : categorical data
    - **quantity** : numerical data

- **teacher_number_of_previously_posted_projects** : numerical data
- **price** : numerical data
- **sentiment score's of each of the essay** : numerical data
- **number of words in the title** : numerical data
- **number of words in the combine essays** : numerical data

And apply the Logistic regression on these features by finding the best hyper paramter as suggested in step 2 and step 3

6. **Conclusion**

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

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakage, 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.

2. Logistic Regression

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

In [32]:

```
cols = list(project_data.columns.values) #Make a list of all of the columns in the df
cols.pop(cols.index('project_is_approved')) #Remove project_is_approved from list
project_data_LR = project_data[cols+['project_is_approved']]
project_data_LR.head(2)
```

Out[32]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_title	proje
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My Eng tha
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	(sch

2 rows × 22 columns

In [33]:

```
#project_data_LR=project_data_LR[0:50000]
X=np.array(project_data_LR.iloc[:,0:21])
Y=np.array(project_data_LR['project_is_approved'])
```

In [34]:

```
print(X.shape)
print(Y.shape)
```

```
(109248, 21)
(109248,)
```

In [35]:

```
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)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33)
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
```

```
(49041, 21) (49041,)
(24155, 21) (24155,)
(36052, 21) (36052,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [36]:

```
project_data_LR.columns
```

Out[36]:

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

Vectorizing Clean Categories

In [37]:

```
vectorizer = CountVectorizer(lowercase=False)
features_set1=[]
features_set2=[]
vectorizer.fit(X_train[:,13])
X_train_clean_categories = vectorizer.transform(X_train[:,13])
X_cv_clean_categories = vectorizer.transform(X_cv[:,13])
X_test_clean_categories = vectorizer.transform(X_test[:,13])

print("After vectorizations")
print(X_train_clean_categories.shape, y_train.shape)
print(X_cv_clean_categories.shape, y_cv.shape)
print(X_test_clean_categories.shape, y_test.shape)
features_set1.extend(vectorizer.get_feature_names())
features_set2.extend(vectorizer.get_feature_names())
print(features_set1)
print(features_set2)
```

```
After vectorizations
(49041, 9) (49041,)
(24155, 9) (24155,)
(36052, 9) (36052,)
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
```

Vectorizing Clean Sub Categories

In [38]:

```
vectorizer = CountVectorizer(lowercase=False)
vectorizer.fit(X_train[:,14])
X_train_clean_subcategories = vectorizer.transform(X_train[:,14])
X_train_clean_subcategories = vectorizer.transform(X_train[:,14])
X_cv_clean_subcategories = vectorizer.transform(X_cv[:,14])
X_test_clean_subcategories = vectorizer.transform(X_test[:,14])

print("After vectorizations")
print(X_train_clean_subcategories.shape, y_train.shape)
print(X_cv_clean_subcategories.shape, y_cv.shape)
print(X_test_clean_subcategories.shape, y_test.shape)
print(vectorizer.get_feature_names())
features_set1.extend(vectorizer.get_feature_names())
features_set2.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 30) (49041,)
(24155, 30) (24155,)
(36052, 30) (36052,)
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
```

Vectorizing School State

In [39]:

```
vectorizer = CountVectorizer(lowercase=False)
vectorizer.fit(X_train[:,4])
X_train_school_state = vectorizer.transform(X_train[:,4])
X_cv_school_state = vectorizer.transform(X_cv[:,4])
X_test_school_state = vectorizer.transform(X_test[:,4])

print("After vectorizations")
print(X_train_school_state.shape, y_train.shape)
print(X_cv_school_state.shape, y_cv.shape)
print(X_test_school_state.shape, y_test.shape)
print(vectorizer.get_feature_names())
features_set1.extend(vectorizer.get_feature_names())
features_set2.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
```

Vectorizing Teacher Prefix

In [40]:

```
vectorizer = CountVectorizer(lowercase=False)
vectorizer.fit(X_train[:,3])
X_train_teacher_prefix = vectorizer.transform(X_train[:,3])
X_cv_teacher_prefix = vectorizer.transform(X_cv[:,3])
X_test_teacher_prefix = vectorizer.transform(X_test[:,3])
print("After vectorizations")
print(X_train_teacher_prefix.shape, y_train.shape)
print(X_cv_teacher_prefix.shape, y_cv.shape)
print(X_test_teacher_prefix.shape, y_test.shape)
print(vectorizer.get_feature_names())
features_set1.extend(vectorizer.get_feature_names())
features_set2.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 5) (49041,)
(24155, 5) (24155,)
(36052, 5) (36052,)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
```

Vectorizing Project Grade Category

In [41]:

```
vectorizer = CountVectorizer(lowercase=False)
vectorizer.fit(X_train[:,18])
X_train_grades = vectorizer.transform(X_train[:,18])
X_cv_grades = vectorizer.transform(X_cv[:,18])
X_test_grades = vectorizer.transform(X_test[:,18])
print("After vectorizations")
print(X_train_grades.shape, y_train.shape)
print(X_cv_grades.shape, y_cv.shape)
print(X_test_grades.shape, y_test.shape)
print(vectorizer.get_feature_names())
features_set1.extend(vectorizer.get_feature_names())
features_set2.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 4) (49041,)
(24155, 4) (24155,)
(36052, 4) (36052,)
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
```

Standardizing Prices

In [42]:

```
from sklearn.preprocessing import Normalizer

price_scalar = Normalizer()
price_scalar.fit(X_train[:,16].reshape(-1,1))

X_train_price_standardized_1 = price_scalar.transform(X_train[:,16].reshape(1,-1))
X_train_price_standardized=X_train_price_standardized_1.reshape(-1,1)

X_cv_price_standardized_1 = price_scalar.transform(X_cv[:,16].reshape(1,-1))
X_cv_price_standardized = X_cv_price_standardized_1.reshape(-1,1)

X_test_price_standardized_1 = price_scalar.transform(X_test[:,16].reshape(1,-1))
X_test_price_standardized = X_test_price_standardized_1.reshape(-1,1)

print("After vectorizations")
print(X_train_price_standardized.shape, y_train.shape)
print(X_cv_price_standardized.shape, y_cv.shape)
print(X_test_price_standardized.shape, y_test.shape)

features_set1.extend(['price'])
features_set2.extend(['price'])
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

Standardizing Previous submitted projects

In [43]:

```
from sklearn.preprocessing import Normalizer

prev_projects_scalar = Normalizer()
prev_projects_scalar.fit(X_train[:,12].reshape(-1,1))

X_train_prev_projects_standardized_1 = prev_projects_scalar.transform(X_train[:,12].reshape(1,-1))
X_train_prev_projects_standardized = X_train_prev_projects_standardized_1.reshape(-1,1)

X_cv_prev_projects_standardized_1 = prev_projects_scalar.transform(X_cv[:,12].reshape(1,-1))
```

```
X_cv_prev_projects_standardized_1 = prev_projects_scalar.transform(X_cv[:,12].reshape(1,-1))
X_cv_prev_projects_standardized = X_cv_prev_projects_standardized_1.reshape(-1,1)

X_test_prev_projects_standardized_1 = prev_projects_scalar.transform(X_test[:,12].reshape(1,-1))
X_test_prev_projects_standardized = X_test_prev_projects_standardized_1.reshape(-1,1)

print("After vectorizations")
print(X_train_prev_projects_standardized.shape, y_train.shape)
print(X_cv_prev_projects_standardized.shape, y_cv.shape)
print(X_test_prev_projects_standardized.shape, y_test.shape)
print(X_train_prev_projects_standardized)
features_set1.extend(['previous submitted projects'])
features_set2.extend(['previous submitted projects'])
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
[[0.        ]
 [0.00074777]
 [0.        ]
 ...
 [0.00014955]
 [0.00119644]
 [0.00014955]]
```

2.3 Make Data Model Ready: encoding eassay, and project_title

BOW

In [44]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(ngram_range=(1, 2),min_df=10,max_features=5000)
vectorizer.fit(X_train[:,19])
X_train_preprocessed_essays_bow = vectorizer.transform(X_train[:,19])
X_cv_preprocessed_essays_bow = vectorizer.transform(X_cv[:,19])
X_test_preprocessed_essays_bow = vectorizer.transform(X_test[:,19])
print("After vectorizations")
print(X_train_preprocessed_essays_bow.shape, y_train.shape)
print(X_cv_preprocessed_essays_bow.shape, y_cv.shape)
print(X_test_preprocessed_essays_bow.shape, y_test.shape)
features_set1.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
```

In [45]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(X_train[:,20])
X_train_preprocessed_titles_bow = vectorizer.transform(X_train[:,20])
X_cv_preprocessed_titles_bow = vectorizer.transform(X_cv[:,20])
X_test_preprocessed_titles_bow = vectorizer.transform(X_test[:,20])
print("After vectorizations")
print(X_train_preprocessed_titles_bow.shape, y_train.shape)
print(X_cv_preprocessed_titles_bow.shape, y_cv.shape)
print(X_test_preprocessed_titles_bow.shape, y_test.shape)
features_set1.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 2014) (49041,)
(24155, 2014) (24155,)
(36052, 2014) (36052,)
```

TFIDF

In [46]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1, 2),min_df=10,max_features=5000)
vectorizer.fit(X_train[:,19])
X_train_preprocessed_essays_tfidf = vectorizer.transform(X_train[:,19])
X_cv_preprocessed_essays_tfidf = vectorizer.transform(X_cv[:,19])
X_test_preprocessed_essays_tfidf = vectorizer.transform(X_test[:,19])
print("After vectorizations")
print(X_train_preprocessed_essays_tfidf.shape, y_train.shape)
print(X_cv_preprocessed_essays_tfidf.shape, y_cv.shape)
print(X_test_preprocessed_essays_tfidf.shape, y_test.shape)
features_set2.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
```

In [47]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(X_train[:,20])
X_train_preprocessed_titles_tfidf = vectorizer.transform(X_train[:,20])
X_cv_preprocessed_titles_tfidf = vectorizer.transform(X_cv[:,20])
X_test_preprocessed_titles_tfidf = vectorizer.transform(X_test[:,20])
print("After vectorizations")
print(X_train_preprocessed_titles_tfidf.shape, y_train.shape)
print(X_cv_preprocessed_titles_tfidf.shape, y_cv.shape)
print(X_test_preprocessed_titles_tfidf.shape, y_test.shape)
features_set2.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(49041, 2014) (49041,)
(24155, 2014) (24155,)
(36052, 2014) (36052,)
```

Avg W2V

In [48]:

```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [49]:

```
X_train_preprocessed_essays_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored
in this list
for sentence in tqdm(X_train[:,19]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_train_preprocessed_essays_avg_w2v_vectors.append(vector)

print(len(X_train_preprocessed_essays_avg_w2v_vectors))
print(len(X_train_preprocessed_essays_avg_w2v_vectors[0]))

X_cv_preprocessed_essays_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(X_cv[:,19]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
    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
    X_cv_preprocessed_essays_avg_w2v_vectors.append(vector)

print(len(X_cv_preprocessed_essays_avg_w2v_vectors))
print(len(X_cv_preprocessed_essays_avg_w2v_vectors[0]))

X_test_preprocessed_essays_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored
in this list
for sentence in tqdm(X_test[:,19]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_test_preprocessed_essays_avg_w2v_vectors.append(vector)

print(len(X_test_preprocessed_essays_avg_w2v_vectors))
print(len(X_test_preprocessed_essays_avg_w2v_vectors[0]))
```

```
100%|██████████████████████████████████| 49041/49041 [00:22<00:00, 2144.03it/s]
```

```
49041
300
```

```
100%|██████████████████████████████████| 24155/24155 [00:11<00:00, 2147.19it/s]
```

```
24155
300
```

```
100%|██████████████████████████████████| 36052/36052 [00:17<00:00, 2060.98it/s]
```

```
36052
300
```

In [50]:

```
X_train_preprocessed_titles_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored
in this list
for sentence in tqdm(X_train[:,20]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_train_preprocessed_titles_avg_w2v_vectors.append(vector)

print(len(X_train_preprocessed_titles_avg_w2v_vectors))
print(len(X_train_preprocessed_titles_avg_w2v_vectors[0]))

X_cv_preprocessed_titles_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(X_cv[:,20]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_cv_preprocessed_titles_avg_w2v_vectors.append(vector)
```

```
print(len(X_cv_preprocessed_titles_avg_w2v_vectors))
print(len(X_cv_preprocessed_titles_avg_w2v_vectors[0]))

X_test_preprocessed_titles_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test[:,20]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    X_test_preprocessed_titles_avg_w2v_vectors.append(vector)

print(len(X_test_preprocessed_titles_avg_w2v_vectors))
print(len(X_test_preprocessed_titles_avg_w2v_vectors[0]))
```

```
100%|█████████████████████████████████| 49041/49041 [00:01<00:00, 33820.74it/s]
```

```
49041
300
```

```
100%|█████████████████████████████████| 24155/24155 [00:00<00:00, 32442.43it/s]
```

```
24155
300
```

```
100%|█████████████████████████████████| 36052/36052 [00:00<00:00, 39459.39it/s]
```

```
36052
300
```

TFIDF W2V For Essay

In [51]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train[:,19])
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [52]:

```
# average Word2Vec
# compute average word2vec for each review.
X_train_preprocessed_essays_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train[:,19]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_preprocessed_essays_tfidf_w2v_vectors.append(vector)

print(len(X_train_preprocessed_essays_tfidf_w2v_vectors))
print(len(X_train_preprocessed_essays_tfidf_w2v_vectors[0]))

# average Word2Vec
# compute average word2vec for each review.
X_cv_preprocessed_essays_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored
```

```
in this list
for sentence in tqdm(X_cv[:,19]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_cv_preprocessed_essays_tfidf_w2v_vectors.append(vector)

print(len(X_cv_preprocessed_essays_tfidf_w2v_vectors))
print(len(X_cv_preprocessed_essays_tfidf_w2v_vectors[0]))

# average Word2Vec
# compute average word2vec for each review.
X_test_preprocessed_essays_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm(X_test[:,19]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_preprocessed_essays_tfidf_w2v_vectors.append(vector)

print(len(X_test_preprocessed_essays_tfidf_w2v_vectors))
print(len(X_test_preprocessed_essays_tfidf_w2v_vectors[0]))
```

```
100%|███████████████████████████████████| 49041/49041 [02:39<00:00, 308.19it/s]
```

```
49041
300
```

```
100%|███████████████████████████████████| 24155/24155 [01:16<00:00, 316.12it/s]
```

```
24155
300
```

```
100%|███████████████████████████████████| 36052/36052 [01:55<00:00, 312.09it/s]
```

```
36052
300
```

TFIDF W2V For Titles

In [53]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train[:,20])
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [54]:

```
# average Word2Vec
# compute average word2vec for each review.
X_train_preprocessed_titles_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stor
ed in this list
for sentence in tqdm(X_train[:,20]): # for each review/sentence
```

```
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_preprocessed_titles_tfidf_w2v_vectors.append(vector)

print(len(X_train_preprocessed_titles_tfidf_w2v_vectors))
print(len(X_train_preprocessed_titles_tfidf_w2v_vectors[0]))

# average Word2Vec
# compute average word2vec for each review.
X_cv_preprocessed_titles_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored
in this list
for sentence in tqdm(X_cv[:,20]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_cv_preprocessed_titles_tfidf_w2v_vectors.append(vector)

print(len(X_cv_preprocessed_titles_tfidf_w2v_vectors))
print(len(X_cv_preprocessed_titles_tfidf_w2v_vectors[0]))

# average Word2Vec
# compute average word2vec for each review.
X_test_preprocessed_titles_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm(X_test[:,20]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_preprocessed_titles_tfidf_w2v_vectors.append(vector)

print(len(X_test_preprocessed_titles_tfidf_w2v_vectors))
print(len(X_test_preprocessed_titles_tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████| 49041/49041 [00:02<00:00, 18534.87it/s]
```

```
49041
300
```

```
100%|██████████████████████████████████| 24155/24155 [00:01<00:00, 17696.37it/s]
```

```
24155
300
```

```
100%|██████████████████████████████████| 36052/36052 [00:01<00:00, 18418.20it/s]
```

```
36052
300
```

SET 1-- BOW

In [55]:

```
from scipy.sparse import hstack
X1_train = hstack((X_train_clean_categories,
X_train_clean_subcategories,X_train_school_state,X_train_teacher_prefix,X_train_grades,X_train_pric
e_standardized,X_train_prev_projects_standardized,X_train_preprocessed_essays_bow,X_train_preproces
sed_titles_bow))
X1_train.shape
```

Out[55]:

```
(49041, 7115)
```

In [56]:

```
X1_cv = hstack((X_cv_clean_categories,
X_cv_clean_subcategories,X_cv_school_state,X_cv_teacher_prefix,X_cv_grades,X_cv_price_standardized
,X_cv_prev_projects_standardized,X_cv_preprocessed_essays_bow,X_cv_preprocessed_titles_bow))
X1_cv.shape
```

Out[56]:

```
(24155, 7115)
```

In [57]:

```
X1_test = hstack((X_test_clean_categories,
X_test_clean_subcategories,X_test_school_state,X_test_teacher_prefix,X_test_grades,X_test_price_sta
ndardized,X_test_prev_projects_standardized,X_test_preprocessed_essays_bow,X_test_preprocessed_titl
es_bow))
X1_test.shape
```

Out[57]:

```
(36052, 7115)
```

SET 2--TFIDF

In [58]:

```
from scipy.sparse import hstack
X2_train = hstack((X_train_clean_categories,
X_train_clean_subcategories,X_train_school_state,X_train_teacher_prefix,X_train_grades,X_train_pric
e_standardized,X_train_prev_projects_standardized,X_train_preprocessed_essays_tfidf,X_train_preproc
essed_titles_tfidf))
X2_train.shape
```

Out[58]:

```
(49041, 7115)
```

In [59]:

```
X2_cv = hstack((X_cv_clean_categories,
X_cv_clean_subcategories,X_cv_school_state,X_cv_teacher_prefix,X_cv_grades,X_cv_price_standardized
,X_cv_prev_projects_standardized,X_cv_preprocessed_essays_tfidf,X_cv_preprocessed_titles_tfidf))
X2_cv.shape
```

Out[59]:

```
(24155, 7115)
```

In [60]:

```
X2_test = hstack((X_test_clean_categories,
X2_test = hstack((X_test_clean_categories,
X_test_clean_subcategories,X_test_school_state,X_test_teacher_prefix,X_test_grades,X_test_price_sta
ndardized,X_test_prev_projects_standardized,X_test_preprocessed_essays_tfidf,X_test_preprocessed_ti
tles_tfidf))
X2_test.shape
```

Out[60]:

```
(36052, 7115)
```

SET 3--AVGW2V

In [61]:

```
X3_train = hstack((X_train_clean_categories,
X_train_clean_subcategories,X_train_school_state,X_train_teacher_prefix,X_train_grades,X_train_pric
e_standardized,X_train_prev_projects_standardized,X_train_preprocessed_essays_avg_w2v_vectors,X_tr
ain_preprocessed_titles_avg_w2v_vectors))
X3_train.shape
```

Out[61]:

```
(49041, 701)
```

In [62]:

```
X3_cv = hstack((X_cv_clean_categories,
X_cv_clean_subcategories,X_cv_school_state,X_cv_teacher_prefix,X_cv_grades,X_cv_price_standardized
,X_cv_prev_projects_standardized,X_cv_preprocessed_essays_avg_w2v_vectors,X_cv_preprocessed_titles_
avg_w2v_vectors))
X3_cv.shape
```

Out[62]:

```
(24155, 701)
```

In [63]:

```
X3_test = hstack((X_test_clean_categories,
X_test_clean_subcategories,X_test_school_state,X_test_teacher_prefix,X_test_grades,X_test_price_sta
ndardized,X_test_prev_projects_standardized,X_test_preprocessed_essays_avg_w2v_vectors,X_test_prepr
ocessed_titles_avg_w2v_vectors))
X3_test.shape
```

Out[63]:

```
(36052, 701)
```

SET 4--TFIDFW2V

In [64]:

```
X4_train = hstack((X_train_clean_categories,
X_train_clean_subcategories,X_train_school_state,X_train_teacher_prefix,X_train_grades,X_train_pric
e_standardized,X_train_prev_projects_standardized,X_train_preprocessed_essays_tfidf_w2v_vectors,X_
train_preprocessed_titles_tfidf_w2v_vectors))
X4_train.shape
```

Out[64]:

```
(49041, 701)
```

In [65]:

```
X4_cv = hstack((X_cv_clean_categories,
X_cv_clean_subcategories,X_cv_school_state,X_cv_teacher_prefix,X_cv_grades,X_cv_price_standardized
,X_cv_prev_projects_standardized,X_cv_preprocessed_essays_tfidf_w2v_vectors,X_cv_preprocessed_title
s_tfidf_w2v_vectors))
```

```
X4_cv.shape
```

Out[65]:

```
(24155, 701)
```

In [66]:

```
X4_test = hstack((X_test_clean_categories,
X_test_clean_subcategories,X_test_school_state,X_test_teacher_prefix,X_test_grades,X_test_price_sta
ndardized,X_test_prev_projects_standardized,X_test_preprocessed_essays_tfidf_w2v_vectors,X_test_pre
processed_titles_tfidf_w2v_vectors))
X4_test.shape
```

Out[66]:

```
(36052, 701)
```

SET 5

Title Word Count

In [67]:

```
title_words_counts_train=[]
for sent in X_train[:,20]:
    l=len(sent.split())
    title_words_counts_train.append(l)

title_words_counts_cv=[]
for sent in X_cv[:,20]:
    l=len(sent.split())
    title_words_counts_cv.append(l)

title_words_counts_test=[]
for sent in X_test[:,20]:
    l=len(sent.split())
    title_words_counts_test.append(l)
```

Essay Word Count

In [68]:

```
essay_words_counts_train=[]
for sent in X_train[:,19]:
    l=len(sent.split())
    essay_words_counts_train.append(l)

essay_words_counts_cv=[]
for sent in X_cv[:,19]:
    l=len(sent.split())
    essay_words_counts_cv.append(l)

essay_words_counts_test=[]
for sent in X_test[:,19]:
    l=len(sent.split())
    essay_words_counts_test.append(l)
```

In [69]:

```
analyser = SentimentIntensityAnalyzer()

neg_train = []
pos_train = []
neu_train = []
compound_train = []
for a in (X_train[:,19]) :
    b = analyser.polarity_scores(a)['neg']
    c = analyser.polarity_scores(a)['pos']
    d = analyser.polarity_scores(a)['neu']
    e = analyser.polarity_scores(a)['compound']
    neg_train.append(b)
    pos_train.append(c)
    neu_train.append(d)
    compound_train.append(e)
```

In [70]:

```
neg_train_ar=np.asarray(neg_train)
pos_train_ar=np.asarray(pos_train)
neu_train_ar=np.asarray(neu_train)
compound_train_ar=np.asarray(compound_train)
```

In [71]:

```
analyser = SentimentIntensityAnalyzer()

neg_cv = []
pos_cv = []
neu_cv = []
compound_cv = []
for a in tqdm(X_cv[:,19]) :
    b = analyser.polarity_scores(a)['neg']
    c = analyser.polarity_scores(a)['pos']
    d = analyser.polarity_scores(a)['neu']
    e = analyser.polarity_scores(a)['compound']
    neg_cv.append(b)
    pos_cv.append(c)
    neu_cv.append(d)
    compound_cv.append(e)
```

```
100%|████████████████████████████████████| 24155/24155 [04:28<00:00, 89.98it/s]
```

In [72]:

```
neg_cv_ar=np.asarray(neg_cv)
pos_cv_ar=np.asarray(pos_cv)
neu_cv_ar=np.asarray(neu_cv)
compound_cv_ar=np.asarray(compound_cv)
```

In [73]:

```
analyser = SentimentIntensityAnalyzer()

neg_test = []
pos_test = []
neu_test = []
compound_test = []
for a in tqdm(X_test[:,19]) :
    b = analyser.polarity_scores(a)['neg']
    c = analyser.polarity_scores(a)['pos']
    d = analyser.polarity_scores(a)['neu']
    e = analyser.polarity_scores(a)['compound']
    neg_test.append(b)
    pos_test.append(c)
    neu_test.append(d)
    compound_test.append(e)
```

```
100%|████████████████████████████████████| 36052/36052 [06:51<00:00, 87.52it/s]
```

In [74]:

```
neg_test_ar=np.asarray(neg_test)
pos_test_ar=np.asarray(pos_test)
neu_test_ar=np.asarray(neu_test)
compound_test_ar=np.asarray(compound_test)
```

In [75]:

In [75]:

```
essay_words_counts_train_ar=np.asarray(essay_words_counts_train)
title_words_counts_train_ar=np.asarray(title_words_counts_train)

essay_words_counts_cv_ar=np.asarray(essay_words_counts_cv)
title_words_counts_cv_ar=np.asarray(title_words_counts_cv)

essay_words_counts_test_ar=np.asarray(essay_words_counts_test)
title_words_counts_test_ar=np.asarray(title_words_counts_test)
```

In [76]:

```
essay_words_counts_cv_ar.shape

title_words_counts_cv_ar.shape
```

Out[76]:

```
(24155,)
```

In [77]:

```
X5_train = hstack((X_train_clean_categories,
X_train_clean_subcategories,X_train_school_state,X_train_teacher_prefix,X_train_grades,X_train_pric
e_standardized,X_train_prev_projects_standardized,neg_train_ar.reshape(-1,1),pos_train_ar.reshape(
-1,1),neu_train_ar.reshape(-1,1),compound_train_ar.reshape(-1,1),title_words_counts_train_ar.reshap
e(-1,1),essay_words_counts_train_ar.reshape(-1,1)))
X5_train.shape
```

Out[77]:

```
(49041, 107)
```

In [78]:

```
X5_cv = hstack((X_cv_clean_categories,
X_cv_clean_subcategories,X_cv_school_state,X_cv_teacher_prefix,X_cv_grades,X_cv_price_standardized
,X_cv_prev_projects_standardized,neg_cv_ar.reshape(-1,1),pos_cv_ar.reshape(-1,1),neu_cv_ar.reshape
(-1,1),compound_cv_ar.reshape(-1,1),title_words_counts_cv_ar.reshape(-1,1),essay_words_counts_cv_ar
.reshape(-1,1)))
X5_cv.shape
```

Out[78]:

```
(24155, 107)
```

In [79]:

```
X5_test = hstack((X_test_clean_categories,
X_test_clean_subcategories,X_test_school_state,X_test_teacher_prefix,X_test_grades,X_test_price_sta
ndardized,X_test_prev_projects_standardized,neg_test_ar.reshape(-1,1),pos_test_ar.reshape(-1,1),neu
_test_ar.reshape(-1,1),compound_test_ar.reshape(-1,1),title_words_counts_test_ar.reshape(-1,1),essa
y_words_counts_test_ar.reshape(-1,1)))
X5_test.shape
```

Out[79]:

```
(36052, 107)
```

2.4 Appling Logistic Regression on different kind of featurization as mentioned in the instructions

Apply Logistic Regression 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

In [80]:

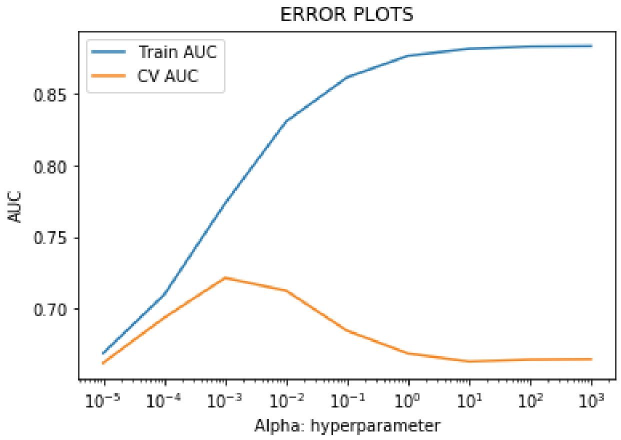
```
def find_best_threshold(threshould, fpr, tpr):
```

```
def find_best_threshould(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

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

2.4.1 Applying Logistic Reggression on BOW, SET 1

In [81]:

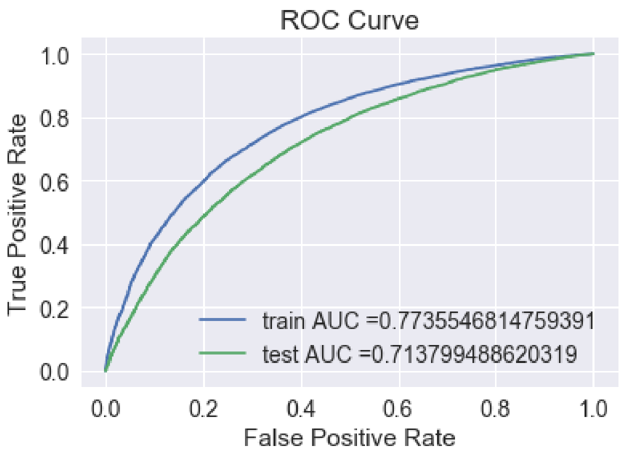
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

train_auc = []
cv_auc = []
alpha = [0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]
for i in alpha:
    neigh = LogisticRegression(penalty='l2',C=i,class_weight='balanced')
    neigh.fit(X1_train, y_train)

    X1_train_csr=X1_train.tocsr()
    X1_cv_csr=X1_cv.tocsr()

    y_train_pred=neigh.predict_proba(X1_train_csr)[:,1]
    y_cv_pred=neigh.predict_proba(X1_cv_csr)[:,1]

    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



In [102]:

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


neigh = LogisticRegression(C=0.001,penalty='l2',class_weight='balanced')
neigh.fit(X1_train, y_train)
```

```
X1_test_csr=X1_test.tocsr()

y_train_pred = neigh.predict(X1_train_csr)
y_test_pred = neigh.predict(X1_test_csr)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, neigh.predict_proba(X1_train_csr)[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, neigh.predict_proba(X1_test_csr)[:,1])

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()

print("="*100)

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```



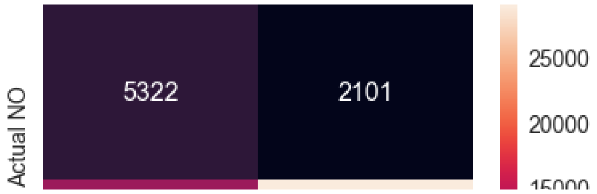
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.5010344238880928 for threshold 0.495
Train confusion matrix
[[ 5322  2101]
 [12579 29039]]
Test confusion matrix
[[ 3438  1986]
 [ 9566 21062]]
```

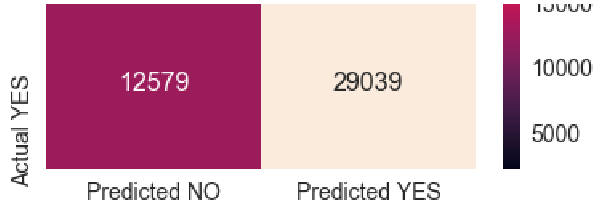
In [103]:

```
import seaborn as sns
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[103]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x71c8bbf60>
```



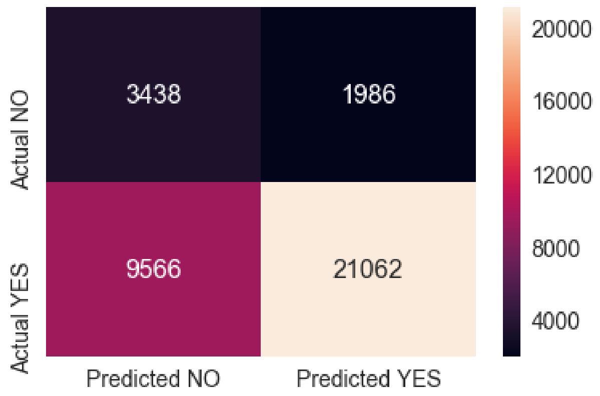


In [104]:

```
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[104]:

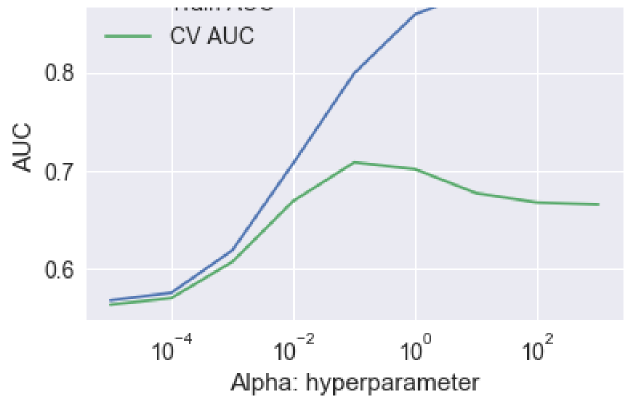
```
<matplotlib.axes._subplots.AxesSubplot at 0x71c7c67b8>
```



2.4.2 Applying Logistic Regression on TFIDF, SET 2

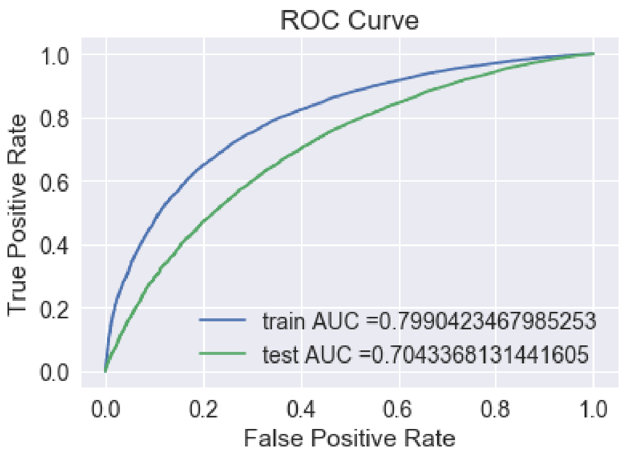
In [85]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

train_auc = []
cv_auc = []
alpha = [0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]
for i in alpha:
    neigh = LogisticRegression(penalty='l2',C=i,class_weight='balanced')
    neigh.fit(X2_train, y_train)

    X2_train_csr=X2_train.tocsr()
    X2_cv_csr=X2_cv.tocsr()

    y_train_pred=neigh.predict_proba(X2_train_csr)[:,1]
    y_cv_pred=neigh.predict_proba(X2_cv_csr)[:,1]

    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



In [105]:

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


neigh = LogisticRegression(C=0.1,penalty='l2',class_weight='balanced')
neigh.fit(X2_train, y_train)

X2_test_csr=X2_test.tocsr()

y_train_pred = neigh.predict(X2_train_csr)
y_test_pred = neigh.predict(X2_test_csr)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, neigh.predict_proba(X2_train_csr)[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, neigh.predict_proba(X2_test_csr)[:,1])

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()

print("="*100)

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```



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

the maximum value of tpr*(1-fpr) 0.5315284625876946 for threshold 0.498
Train confusion matrix
[[ 5521  1902]
 [11931 29687]]
```

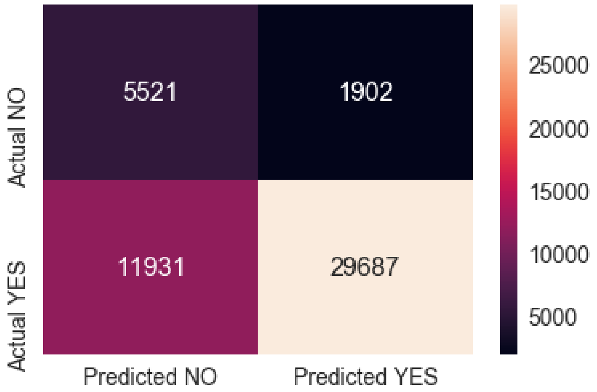
```
Test confusion matrix
[[ 3282  2142]
 [ 9358 21270]]
```

In [106]:

```
import seaborn as sns
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[106]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x71c7c1828>
```

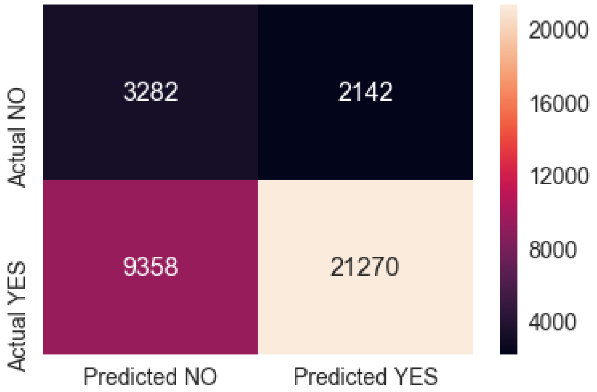


In [107]:

```
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[107]:

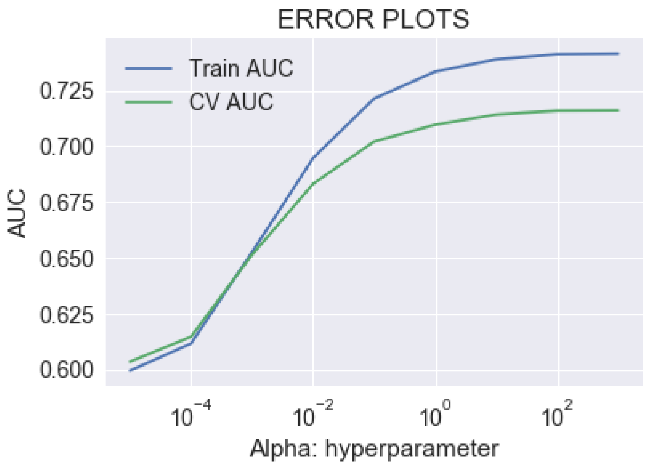
```
<matplotlib.axes._subplots.AxesSubplot at 0x719e0ca58>
```



2.4.3 Applying Logistic Reggression on AVG W2V, SET 3

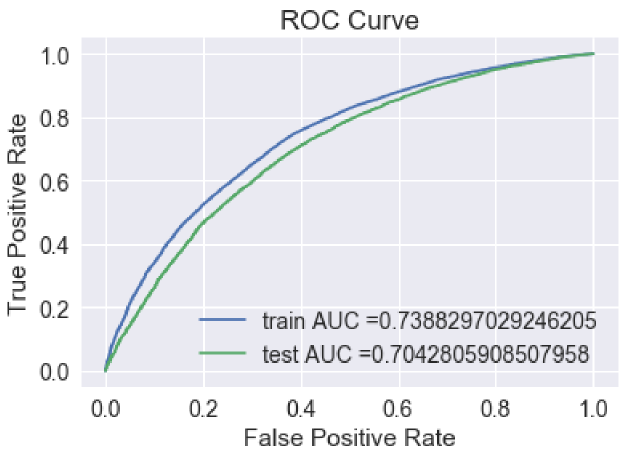
In [89]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

train_auc = []
cv_auc = []
alpha = [0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]
alpha = [0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]
for i in alpha:
    neigh = LogisticRegression(penalty='l2',C=i,class_weight='balanced')
    neigh.fit(X3_train, y_train)

    X3_train_csr=X3_train.tocsr()
    X3_cv_csr=X3_cv.tocsr()

    y_train_pred=neigh.predict_proba(X3_train_csr)[:,1]
    y_cv_pred=neigh.predict_proba(X3_cv_csr)[:,1]

    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



In [90]:

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


neigh = LogisticRegression(C=10,penalty='l2',class_weight='balanced')
neigh.fit(X3_train, y_train)

X3_test_csr=X3_test.tocsr()

y_train_pred = neigh.predict(X3_train_csr)
y_test_pred = neigh.predict(X3_test_csr)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, neigh.predict_proba(X3_train_csr)[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, neigh.predict_proba(X3_test_csr)[:,1])

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()

print("="*100)

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```



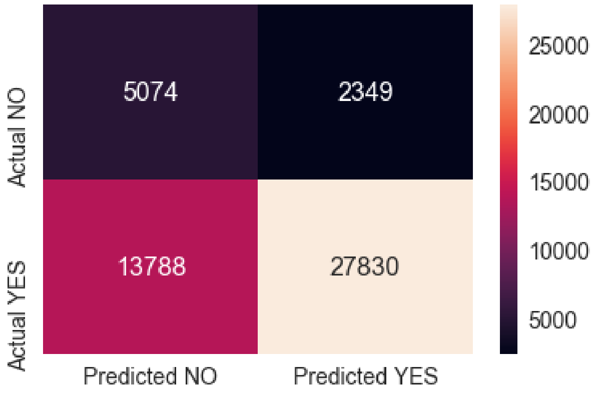
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.46150911188692484 for threshold 0.472
Train confusion matrix
[[ 5074  2349]
 [13788 27830]]
Test confusion matrix
[[ 3494  1930]
 [10287 20341]]
```

In [91]:

```
import seaborn as sns
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x71c8bb588>
```

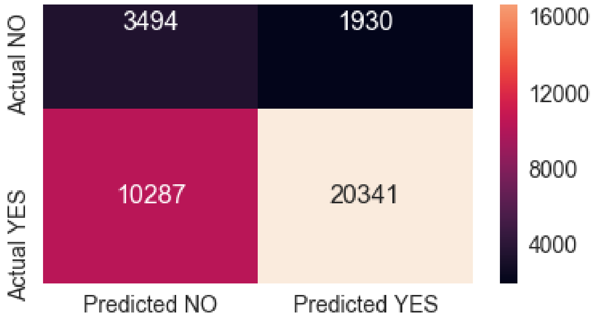


In [92]:

```
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[92]:

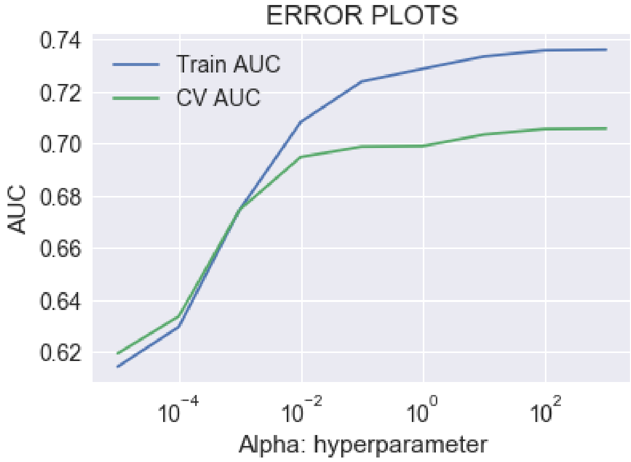
```
<matplotlib.axes._subplots.AxesSubplot at 0x71c84b240>
```



2.4.2 Applying Loggistic Regression on TFIDF W2V, SET 4

In [93]:

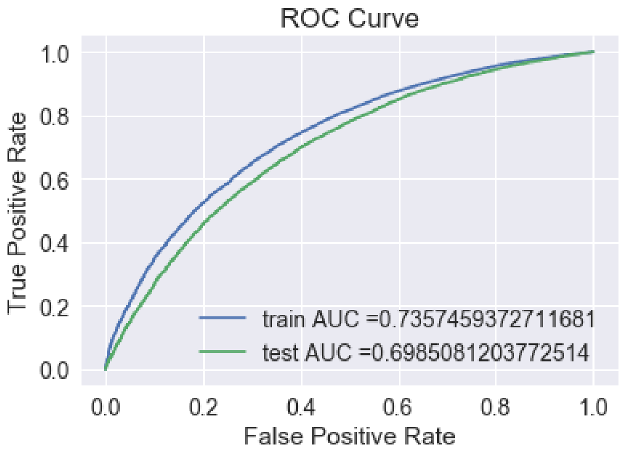
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

train_auc = []
cv_auc = []
alpha = [0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]
for i in alpha:
    neigh = LogisticRegression(penalty='l2',C=i,class_weight='balanced')
    neigh.fit(X4_train, y_train)

    X4_train_csr=X4_train.tocsr()
    X4_cv_csr=X4_cv.tocsr()

    y_train_pred=neigh.predict_proba(X4_train_csr)[:,1]
    y_cv_pred=neigh.predict_proba(X4_cv_csr)[:,1]

    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



In [94]:

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


neigh = LogisticRegression(C=100,penalty='l2',class_weight='balanced')
neigh.fit(X4_train, y_train)
```

```
X4_test_csr=X4_test.tocsr()

y_train_pred = neigh.predict(X4_train_csr)
y_test_pred = neigh.predict(X4_test_csr)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, neigh.predict_proba(X4_train_csr)[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, neigh.predict_proba(X4_test_csr)[:,1])

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()

print("="*100)

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```



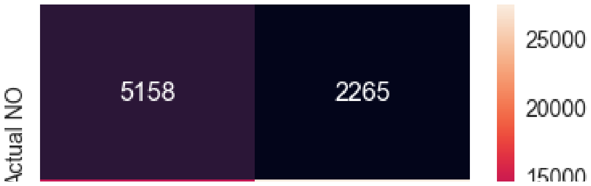
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.4566635320017406 for threshold 0.473
Train confusion matrix
[[ 5158  2265]
 [14333 27285]]
Test confusion matrix
[[ 3541  1883]
 [10937 19691]]
```

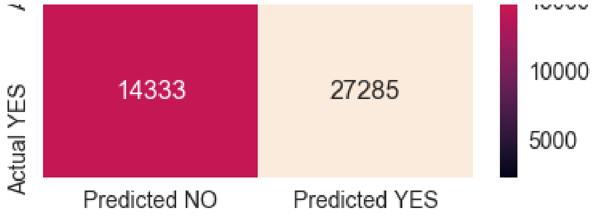
In [95]:

```
import seaborn as sns
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x71a7a90f0>
```



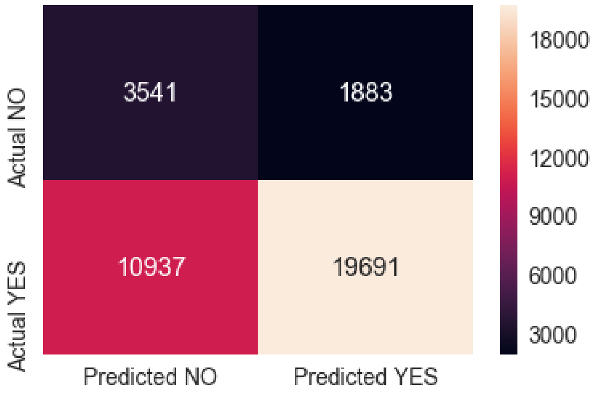


In [96]:

```
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[96]:

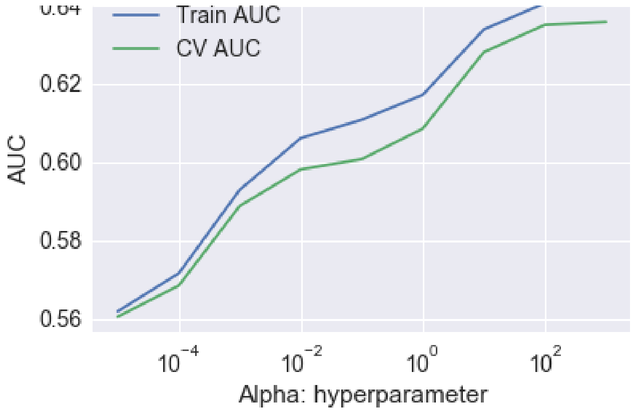
```
<matplotlib.axes._subplots.AxesSubplot at 0x71a8c3240>
```



2.5 Logistic Regression with added Features `Set 5`

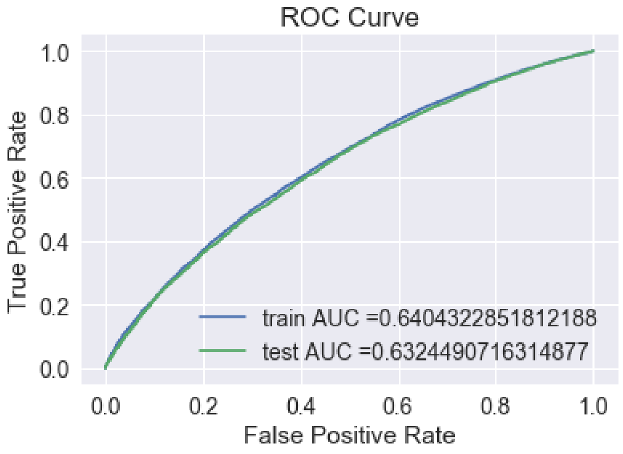
In [97]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt

train_auc = []
cv_auc = []
alpha = [0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000]
for i in alpha:
    neigh = LogisticRegression(penalty='l2',C=i,class_weight='balanced')
    neigh.fit(X5_train, y_train)

    X5_train_csr=X5_train.tocsr()
    X5_cv_csr=X5_cv.tocsr()

    y_train_pred=neigh.predict_proba(X5_train_csr)[:,1]
    y_cv_pred=neigh.predict_proba(X5_cv_csr)[:,1]

    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')
plt.legend()
plt.xlabel("Alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```



In [108]:

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


neigh = LogisticRegression(C=100,penalty='l2',class_weight='balanced')
neigh.fit(X5_train, y_train)

X5_test_csr=X5_test.tocsr()

y_train_pred = neigh.predict(X5_train_csr)
y_test_pred = neigh.predict(X5_test_csr)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, neigh.predict_proba(X5_train_csr)[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, neigh.predict_proba(X5_test_csr)[:,1])

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.show()

print("="*100)

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```



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

the maximum value of tpr*(1-fpr) 0.36086396465969195 for threshold 0.505
Train confusion matrix
[[ 4541  2882]
 [17103 24515]]
```

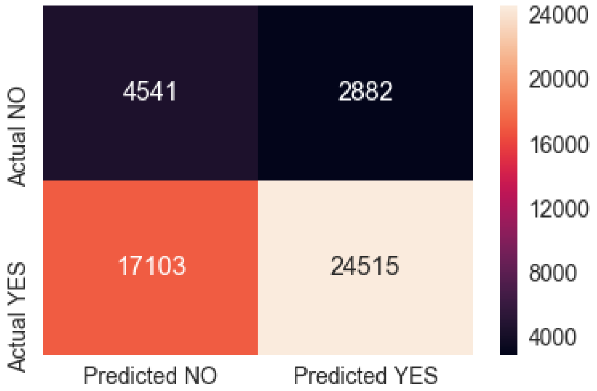
```
Test confusion matrix
[[ 3318  2106]
 [12956 17672]]
```

In [109]:

```
import seaborn as sns
df_cm = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x71c3d4c18>
```

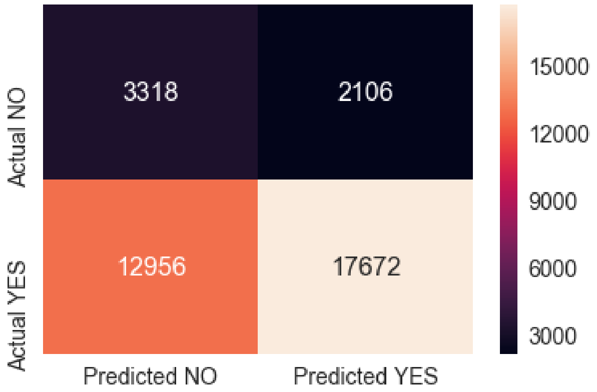


In [110]:

```
df_cm = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
df_cm.columns = ['Predicted NO','Predicted YES']
df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[110]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x71c61ae48>
```



3. Conclusion

In [111]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Model", "HyperParameter", "AUC"]
```

```
x.add_row(["BOW", "Brute Force", 0.001,0.77 ])
x.add_row(["TFIDF", "Brute Force", 0.1,0.79 ])
x.add_row(["AVG W2V", "Brute Force", 10,0.73 ])
x.add_row(["TFIDF W2V", "Brute Force", 100,0.73 ])
x.add_row(["SET 5", "Brute Force", 100,0.64 ])


print(x)
```

```
+------------+-------------+----------------+------+
| Vectorizer |    Model    | HyperParameter | AUC  |
+------------+-------------+----------------+------+
|    BOW     | Brute Force |     0.001      | 0.77 |
|   TFIDF    | Brute Force |      0.1       | 0.79 |
|  AVG W2V   | Brute Force |       10       | 0.73 |
| TFIDF W2V  | Brute Force |      100       | 0.73 |
|   SET 5    | Brute Force |      100       | 0.64 |
+------------+-------------+----------------+------+
```

In [ ]:

```

```