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`

Feature	Description
	• `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*
`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."

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(action='ignore')

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

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

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

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

from tqdm import tqdm
from tqdm import tqdm_notebook
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\bkacharl\AppData\Local\Continuum\anaconda3\lib\site-packages\gensim\utils.py:1197:
UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:

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

In [3]:

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' '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]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[4]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5

In [5]:

```
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[5]:

	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

In [6]:

```
project_grade_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" ", "_")
    project_grade_category.append(a)
```

In [7]:

```
project_grade_category[0:5]
```

Out[7]:

```
['Grades_PreK-2', 'Grades_6-8', 'Grades_6-8', 'Grades_PreK-2', 'Grades_PreK-2']
```

In [8]:

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

project_data["project_grade_category"] = project_grade_category
```

In [9]:

```
project_data.head(5)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_subject_ca
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Math & Science
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Special Needs
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Literacy & Language
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Applied Learning
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Literacy & Language

1.2 preprocessing of `project_subject_categories`

In [10]:

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

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

```
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [11]:

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

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

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

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

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

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

preprocessing of project_grade_category

In [12]:

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

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

grade_cat_list = []
for category in grade_catogories:
    category =str(category)
    category=category.replace('-','_')

    grade_cat_list.append(category.strip())
```

```
project_data['clean_grade_categories'] = grade_cat_list
project_data.drop(['project_grade_category'], 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_grade_categories'].values:
    my_counter.update(word.split())

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

preprocessing of teacher_prefix

In [13]:

```
teacher_prefixes = list(project_data['teacher_prefix'].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

teacher_prefix_list = []
for prefix in teacher_prefixes:
    prefix =str(prefix)
    prefix=prefix.replace('.','')
    teacher_prefix_list.append(prefix.strip())

project_data['clean_teacher_prefix'] = teacher_prefix_list
project_data.drop(['teacher_prefix'], 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_teacher_prefix'].values:
    my_counter.update(word.split())

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

1.4 New feature "Number of Words in Title"

In [14]:

```
title_word_count = []
```

In [15]:

```
for a in project_data["project_title"] :
    b = len(a.split())
    title_word_count.append(b)
```

In [16]:

```
project_data["title_word_count"] = title_word_count
```

In [17]:

```
project_data.head(5)
```

Out[17]:

	Unnamed: 0	id	teacher_id	school_state	Date	project_title	project_essay_1	proj
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	CA	2016-04-27	Engineering STEAM into	I have been fortunate enough	My s com

	Unnamed: 0	id	teacher_id	school_state	Date	project_title	project_essay_1	proj
					00:27:36	the Primary Classroom	to use the Fairy ...	varie back
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagine being 8-9 years old. You're in your th...	Mos stud autis anot
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	CA	2016-04-27 00:46:53	Mobile Learning with a Mobile Listening Center	Having a class of 24 students comes with diver...	I hav twer kind stu..
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	I tea inco scho
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	We publ elem scho

In [18]:

```
# 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 [19]:

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

Out[19]:

	Unnamed: 0	id	teacher_id	school_state	Date	project_title	project_essay_1	proj
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	I have been fortunate enough to use the Fairy ...	My s com varie back
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagine being 8-9 years old. You're in your th...	Mos stud autis anot

In [20]:

```
essay_word_count = []
```

In [21]:

```
for ess in project_data["essay"] :
    c = len(ess.split())
    essay_word_count.append(c)
```

In [22]:

```
project_data["essay_word_count"] = essay_word_count
```

In [23]:

```
project_data.head(5)
```

Out[23]:

	Unnamed: 0	id	teacher_id	school_state	Date	project_title	project_essay_1	proj
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	I have been fortunate enough to use the Fairy ...	My s com varie back
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagine being 8-9 years old. You're in your th...	Mos stud autis anot
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	CA	2016-04-27 00:46:53	Mobile Learning with a Mobile Listening Center	Having a class of 24 students comes with diver...	I hav twer kind stu..
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	I tea inco scho
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	We a publ elem scho

Splitting data into Test - Train

In [24]:

```
# train test split

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data,
project_data['project_is_approved'], test_size=0.33, stratify = project_data['project_is_approved'
])
#X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33,
stratify=y_train)
```

In [25]:

```
X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
#X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

1.3 Text preprocessing

In [26]:

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

In [27]:

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

```
Mrs. Andrews' PreK class is a special group of eager, smart and whimsical children.  \r\nMy class
is full of beautiful children who are experiencing school for the first time, learning to be socia
l, and who are anxious about learning.  \r\nI have 22 students from various backgrounds that does
not inhibit their passion for learning, loving and laughing.  \r\nWe currently have 13 girls and 9
boys who love to express themselves and enjoy life.  Everyday in Mrs. Andrews' class brings a new
adventure and great opportunities to impact their lives.New classroom cubbies will help with our c
lassroom function and flow better.  Currently we store backpacks, coats, lunchboxes and blankets i
n several different locations and it would beneficial to keep them all in one particular place
that's easily accessible to all students.  I would love to have new cubbies to maximize our
learning space.  These materials will help with our classroom routines when unpacking and packing
up at the end of the day.    \r\n\r\nConsider the cost of the cubbies, we don't currently have fun
ds to purchase both furniture and supplies for our classrooms.  It would be great to add these to
our classroom environment.\r\n\r\nnannan
==================================================
Salemwood students are motivated, talented, and some of the most enthusiastic kids I have ever wor
ked with in all my years  of teaching. This is quite heartbreaking as they are also in an
extremely difficult position as we are in a high poverty environment. \r\n\r\nMy students are capa
ble of amazing things and we are developing life long music appreciation and a love of the
performing arts that will last well beyond their middle school years. They are hoping for more opp
ortunities and it is my hope to provide them a chance to be successful in a meaningful
way.\r\n\r\nStudents in band at my school are bright, intuitive, and creative learners. In many ca
ses the best way to learn in music is by doing and my next project with students is for them to le
arn to perform and compose in multiple styles of music. This will allow them to not only
understand HOW to play music, it will give them greater understanding into the idea of WHY we play
music and WHY it is written in specific ways.\r\n\r\nWhether you are listening to the most beautif
ul Rachmaninoff Piano Concerto, an Anime Soundtrack, or  something by Justin Timberlake, there are
reasons behind the sounds on those recordings and logic in the way they are placed in their order
and style. To understand an idea like this, it is important for me to convey ideas of
compositional style and depth to my students. Most of these students are from at risk homes with l
ittle or no money and a free and reduced lunch program. They never have the opportunity outside of
this band room to discover this secret world of music. This opportunity affords students the chanc
e to allow their emotions to flow out in a positive artistic way.\r\n\r\nI cannot tell you how muc
h some of my students have changed since they have begun band. The school now has at least 4 band
performances per year and we have performed everything from the classics to the modern \"popular\"
genre. I have seen students go from needing help to helping others. I have seen the school
community and family grow and blossom into a loving nurturing environment. This project does more
than teach composition and performance...it keeps music in a community that desperately needs it.n
annan
==================================================
My culturally diverse group of students include English language learners, exceptional and gifted
students. My students are required to read daily. Most of the students are from a low economic hou
seholds. They do not have access to any technology outside the school computer lab. My school is f
airly new and is working very hard to provide our students with the highest quality education and
exposure to many developmental experiences that are not available to the students at home. By supp
orting my project you are not only providing my students with the opportunity to explore the world
of reading literacy with integrated technology, but you are also contributing to increasing the se
lf esteem and self worth of numerous students that would not be given such opportunities outside o
f the classroom.\r\nThe items listed in my project will be used during daily instruction. My
students have a difficult time understanding how to edit and respond to specific questions. Using
a document camera, I can easily model and demonstrate how to complete any given task. The students
will see the work being projected on the screen as the corrections are being made in real time. Th
is will also allow the students to remain on task and provide instant feedback on their work. \r\n
\r\nThe implementation of the document camera will allow the students to follow along with the ins
truction being provided as examples are displayed.nannan
```

```
==================================================
I have 34 wonderful 5th grade students, which include special education students, magnet students,
English Language learners, and intervention students.\r\nThey are very enthusiastic about learning
and are always eager to explore new things. They love coming to school every day, and always have
a big smile on their faces, despite the socioeconomic hardships they endure in the community in wh
ich they live. They attend a school in California, in a low income neighborhood. The school is bot
h a regular and highly gifted/magnet institution. However, my students are part of the regular sch
ool, and do not have access to the materials and enrichment activities that the magnet students ha
ve.This project is the result of truly a \"kid-inspired\" idea since my students are very
interested in maintaining an adequate weight and help those students in the class who are obese.
It is for these reasons that my students have inspired me to use technology as a means to stay fit
and healthy and fight childhood obesity.  Their idea of using technology to maintain a healthy
weight and promote fitness in our classroom is by using apps and games online that encourage them
to move around.  They will use the Chromebooks and iPad to visit websites and use apps, such as Tr
ainer, Motion Maze, Kids Yoga Journey, FitQuest, Move Like Me and GoNoodle.  The objective of thes
e games is to get students moving by following what the character is doing on the screen in order
to advance to the next level. This type of exercise will ensure students meet the goal of 60 minut
es of daily physical activity while losing weight and keeping fit.  Students can do the activities
in small groups or I can project the game on a screen so that all students can do the exercises at
the same time.  This type of activity would be great when it is not possible to do P.E. outside
because of hot or rainy weather.\r\nI strongly believe that this 'kid-inspired\" idea is very
important to me as an educator because it will encourage students to develop healthy habits that w
ill stay with them for the rest of their lives.\r\nThank you in advance for your support in
funding our project and making my students' \"kid-inspired\" idea a reality in order to stay fit a
nd healthy and improve their self-esteem.nannan
==================================================
```

In [28]:

```
# 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 [29]:

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

```
I have 34 wonderful 5th grade students, which include special education students, magnet students,
English Language learners, and intervention students.\r\nThey are very enthusiastic about learning
and are always eager to explore new things. They love coming to school every day, and always have
a big smile on their faces, despite the socioeconomic hardships they endure in the community in wh
ich they live. They attend a school in California, in a low income neighborhood. The school is bot
h a regular and highly gifted/magnet institution. However, my students are part of the regular sch
ool, and do not have access to the materials and enrichment activities that the magnet students ha
ve.This project is the result of truly a \"kid-inspired\" idea since my students are very
interested in maintaining an adequate weight and help those students in the class who are obese.
It is for these reasons that my students have inspired me to use technology as a means to stay fit
and healthy and fight childhood obesity.  Their idea of using technology to maintain a healthy
weight and promote fitness in our classroom is by using apps and games online that encourage them
to move around.  They will use the Chromebooks and iPad to visit websites and use apps, such as Tr
ainer, Motion Maze, Kids Yoga Journey, FitQuest, Move Like Me and GoNoodle.  The objective of thes
e games is to get students moving by following what the character is doing on the screen in order
to advance to the next level. This type of exercise will ensure students meet the goal of 60 minut
es of daily physical activity while losing weight and keeping fit.  Students can do the activities
in small groups or I can project the game on a screen so that all students can do the exercises at
the same time.  This type of activity would be great when it is not possible to do P.E. outside
because of hot or rainy weather.\r\nI strongly believe that this 'kid-inspired\" idea is very
```

```
important to me as an educator because it will encourage students to develop healthy habits that w
ill stay with them for the rest of their lives.\r\nThank you in advance for your support in
funding our project and making my students' \"kid-inspired\" idea a reality in order to stay fit a
nd healthy and improve their self-esteem.nannan
==================================================
```

In [30]:

```
# \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)
```

```
I have 34 wonderful 5th grade students, which include special education students, magnet students,
English Language learners, and intervention students.  They are very enthusiastic about learning a
nd are always eager to explore new things. They love coming to school every day, and always have a
big smile on their faces, despite the socioeconomic hardships they endure in the community in
which they live. They attend a school in California, in a low income neighborhood. The school is b
oth a regular and highly gifted/magnet institution. However, my students are part of the regular s
chool, and do not have access to the materials and enrichment activities that the magnet students
have.This project is the result of truly a  kid-inspired  idea since my students are very
interested in maintaining an adequate weight and help those students in the class who are obese.
It is for these reasons that my students have inspired me to use technology as a means to stay fit
and healthy and fight childhood obesity.  Their idea of using technology to maintain a healthy
weight and promote fitness in our classroom is by using apps and games online that encourage them
to move around.  They will use the Chromebooks and iPad to visit websites and use apps, such as Tr
ainer, Motion Maze, Kids Yoga Journey, FitQuest, Move Like Me and GoNoodle.  The objective of thes
e games is to get students moving by following what the character is doing on the screen in order
to advance to the next level. This type of exercise will ensure students meet the goal of 60 minut
es of daily physical activity while losing weight and keeping fit.  Students can do the activities
in small groups or I can project the game on a screen so that all students can do the exercises at
the same time.  This type of activity would be great when it is not possible to do P.E. outside
because of hot or rainy weather.  I strongly believe that this 'kid-inspired  idea is very
important to me as an educator because it will encourage students to develop healthy habits that w
ill stay with them for the rest of their lives.  Thank you in advance for your support in funding
our project and making my students'  kid-inspired  idea a reality in order to stay fit and healthy
and improve their self-esteem.nannan
```

In [31]:

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

```
I have 34 wonderful 5th grade students which include special education students magnet students En
glish Language learners and intervention students They are very enthusiastic about learning and ar
e always eager to explore new things They love coming to school every day and always have a big sm
ile on their faces despite the socioeconomic hardships they endure in the community in which they
live They attend a school in California in a low income neighborhood The school is both a regular
and highly gifted magnet institution However my students are part of the regular school and do not
have access to the materials and enrichment activities that the magnet students have This project
is the result of truly a kid inspired idea since my students are very interested in maintaining an
adequate weight and help those students in the class who are obese It is for these reasons that my
students have inspired me to use technology as a means to stay fit and healthy and fight childhood
obesity Their idea of using technology to maintain a healthy weight and promote fitness in our cla
ssroom is by using apps and games online that encourage them to move around They will use the Chro
mebooks and iPad to visit websites and use apps such as Trainer Motion Maze Kids Yoga Journey FitQ
uest Move Like Me and GoNoodle The objective of these games is to get students moving by following
what the character is doing on the screen in order to advance to the next level This type of
exercise will ensure students meet the goal of 60 minutes of daily physical activity while losing
weight and keeping fit Students can do the activities in small groups or I can project the game on
a screen so that all students can do the exercises at the same time This type of activity would be
great when it is not possible to do P E outside because of hot or rainy weather I strongly believe
that this kid inspired idea is very important to me as an educator because it will encourage stude
nts to develop healthy habits that will stay with them for the rest of their lives Thank you in ad
vance for your support in funding our project and making my students kid inspired idea a reality i
n order to stay fit and healthy and improve their self esteem nannan
```

In [32]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
```

```
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [33]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['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.lower() not in stopwords)
    preprocessed_essays_train.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 73196/73196 [01:
14<00:00, 979.02it/s]
```

In [34]:

```
# after preprocesing
preprocessed_essays_train[100]
```

Out[34]:

```
'large number active second grade students class least class attention deficit hyperactivity
disorder adhd rest normal active 7 8 year olds hard sit still would like provide alternatives trad
itional student desk first year teacher resources limited tall table kids stand work couple floor
pillows lay clipboards write sitting floor tables would like purchase stability balance balls
wiggle seat cushions lap desks students use flexible seating one time research shows flexible seat
ing arrangements many benefits including burning calories using excess energy increased motivation
engagement creating better oxygen flow brain improving core strength overall posture studies also
link physical activity academic performance researchers found physical activity increases concentr
ation mental cognition academic performance children attentive better behaved performed higher sch
olastically anything help students concentrate help education instead students getting walking aro
und room gently bouncing stability ball wiggle seat still concentrating assignment nannan'
```

Preprocessed test data

In [35]:

```
preprocessed_essays_test = []
```

```
# tqdm is for printing the status bar
for sentence in tqdm(X_test['essay'].values):
    sent = decontracted(sentence)
    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.lower() not in stopwords)
    preprocessed_essays_test.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 36052/36052
[00:33<00:00, 1089.41it/s]
```

In [36]:

```
preprocessed_essays_test[1000]
```

Out[36]:

```
'students ap literature class inspire every day curiosity enthusiasm work ethic like support
traits providing engaging books read given day may participate socratic discussions analyze poetry
engage close readings challenging texts visitors enter high school drastically served neighborhood
south side chicago not expect see get past doors school high ceilings bright hallways festooned st
udent artwork studious kids working consulting teachers free hours full library warm atmosphere pr
oud say oasis prompted several students call school classroom home teachers volunteer run programs
organize clubs order maximize students cultural intellectual experiences counselors help students
secure abundant scholarships students strive succeed beyond act sat curious want know create leave
marks world books help students engage analyze classic literature relevant lives novel narrator ho
lden caulfield provides unique youthful worldview students today still relate read explore deeper
meanings prepare ap test real world holden struggles adolescence adulthood mirror many challenges
students face today hope book resonate grow lifelong readers learners reading salinger timeless bi
ldungsroman help students appreciate beauty complexity possibility inherent english language life
nannan'
```

1.4 Preprocessing of `project_title`

In [38]:

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

In [39]:

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

```
Engineering STEAM into the Primary Classroom
==================================================
Building Blocks for Learning
==================================================
Empowering Students Through Art:Learning About Then and Now
==================================================
Health Nutritional Cooking in Kindergarten
==================================================
```

Preprocessing of Project Title for Train data

In [40]:

```
preprocessed_titles_train = []
```

```
for titles in tqdm(X_train["project_title"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f.lower() not in stopwords)
    preprocessed_titles_train.append(title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████| 73196/73196
[00:03<00:00, 23024.89it/s]
```

In [41]:

```
preprocessed_titles_train[1000]
```

Out[41]:

```
'eyes readers'
```

Preprocessing of Project Title for Test data

In [42]:

```
preprocessed_titles_test = []

for titles in tqdm(X_test["project_title"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f.lower() not in stopwords)
    preprocessed_titles_test.append(title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████| 36052/36052
[00:01<00:00, 25903.87it/s]
```

In [43]:

```
preprocessed_titles_test[1000]
```

Out[43]:

```
'books stand test time'
```

Preprocessing of teacher_prefix for train data

In [44]:

```
preprocessed_teachers_prefix_train=[]

for prefix in tqdm(X_train['clean_teacher_prefix']):
    prefix=str(prefix)
    prefix = prefix.replace("."," ")
    preprocessed_teachers_prefix_train.append(prefix.strip())
```

```
100%|████████████████████████████████████████████████████████████| 73196/73196
[00:00<00:00, 1106173.46it/s]
```

Preprocessing of teacher_prefix for test data

In [45]:

```
preprocessed_teachers_prefix_test=[]

for prefix in tqdm(X_test['clean_teacher_prefix']):
    prefix=str(prefix)
    prefix = prefix.replace("."," ")
    preprocessed_teachers_prefix_test.append(prefix.strip())
```

```
100%|████████████████████████████████████████████████████████████| 36052/36052
[00:00<00:00, 967156.91it/s]
```

Preprocessing of project_category for train data

In [46]:

```
preprocessed_project_category_train=[]

for category in tqdm(X_train['clean_grade_categories']):
    category=str(category)
    category = category.replace("-","_")
    preprocessed_project_category_train.append(category.strip())
```

```
100%|████████████████████████████████████████████████████████████| 73196/73196
[00:00<00:00, 1123153.97it/s]
```

Preprocessing of project_category for test data

In [47]:

```
preprocessed_project_category_test=[]

for category in tqdm(X_test['clean_grade_categories']):
    category=str(category)
    category = category.replace("-","_")
    preprocessed_project_category_test.append(category.strip())
```

```
100%|████████████████████████████████████████████████████████████| 36052/36052
[00:00<00:00, 1059204.18it/s]
```

1.5 Preparing data for models

In [48]:

```
project_data.columns
```

Out[48]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state', 'Date',
       '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', 'clean_grade_categories',
       'clean_teacher_prefix', 'title_word_count', 'essay',
       'essay_word_count'],
      dtype='object')
```

we are going to consider

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

- project_title : text data

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

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

1.5.1 Vectorizing Categorical data

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

One hot encode clean_categories

In [49]:

```
# we use count vectorizer to convert the values into one

from sklearn.feature_extraction.text import CountVectorizer

vectorizer_proj = CountVectorizer(lowercase=False, binary=True)
vectorizer_proj.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer_proj.transform(X_train['clean_categories'].values)
categories_one_hot_test = vectorizer_proj.transform(X_test['clean_categories'].values)
#categories_one_hot_cv = vectorizer_proj.transform(X_cv['clean_categories'].values)

print(vectorizer_proj.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",categories_one_hot_test.shape)
#print("Shape of matrix of CV data after one hot encoding ",categories_one_hot_cv.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix of Train data after one hot encoding  (73196, 9)
Shape of matrix of Test data after one hot encoding  (36052, 9)
```

One hot encode clean_subcategories

In [50]:

```
# we use count vectorizer to convert the values into one

vectorizer_sub_proj = CountVectorizer(lowercase=False, binary=True)
vectorizer_sub_proj.fit(X_train['clean_subcategories'].values)

sub_categories_one_hot_train = vectorizer_sub_proj.transform(X_train['clean_subcategories'].values
)
sub_categories_one_hot_test = vectorizer_sub_proj.transform(X_test['clean_subcategories'].values)
#sub_categories_one_hot_cv = vectorizer_sub_proj.transform(X_cv['clean_subcategories'].values)


print(vectorizer_sub_proj.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",sub_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",sub_categories_one_hot_test.shape)
#print("Shape of matrix of Cross Validation data after one hot encoding
",sub_categories_one_hot_cv.shape)
```

```
['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']
Shape of matrix of Train data after one hot encoding  (73196, 30)
Shape of matrix of Test data after one hot encoding  (36052, 30)
```

In [51]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

One hot encode on state

In [52]:

```
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [53]:

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

In [54]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer_states = CountVectorizer(lowercase=False,binary=True)
vectorizer_states.fit(X_train['school_state'].values)

school_state_categories_one_hot_train = vectorizer_states.transform(X_train['school_state'].values
)
school_state_categories_one_hot_test = vectorizer_states.transform(X_test['school_state'].values)
#school_state_categories_one_hot_cv = vectorizer_states.transform(X_cv['school_state'].values)

print(vectorizer_states.get_feature_names())

print("Shape of matrix of Train data after one hot encoding
",school_state_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_categories_one_hot_test.
shape)
#print("Shape of matrix of Cross Validation data after one hot encoding
",school_state_categories_one_hot_cv.shape)
```

```
['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']
Shape of matrix of Train data after one hot encoding  (73196, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
```

One hot encode - project category

In [55]:

```
#my_counter = Counter()
#for project_grade in project_data['project_grade_category'].values:
#    my_counter.update(project_grade.split())
```

In [56]:

```
#project_grade_cat_dict = dict(my_counter)
#sorted_project_grade_cat_dict = dict(sorted(project_grade_cat_dict.items(), key=lambda kv: kv[1])
)
```

In [57]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer_grade = CountVectorizer(lowercase=False, binary=True)
vectorizer_grade.fit(X_train['clean_grade_categories'].values.astype("U"))
```

```
project_grade_categories_one_hot_train =
vectorizer_grade.transform(X_train['clean_grade_categories'].values.astype("U"))
project_grade_categories_one_hot_test = vectorizer_grade.transform(X_test['clean_grade_categories'
].values.astype("U"))
#project_grade_categories_one_hot_cv =
vectorizer_grade.transform(X_cv['clean_grade_categories'].values.astype("U"))

print(vectorizer_grade.get_feature_names())

print("Shape of matrix of Train data after one hot encoding
",project_grade_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_one_hot_test
.shape)
#print("Shape of matrix of Cross Validation data after one hot encoding
",project_grade_categories_one_hot_cv.shape)
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of matrix of Train data after one hot encoding  (73196, 4)
Shape of matrix of Test data after one hot encoding  (36052, 4)
```

One hot encode - Teacher Prefix

In [58]:

```
#my_counter = Counter()
#for teacher_prefix in project_data['teacher_prefix'].values:
#    teacher_prefix = str(teacher_prefix)
#    my_counter.update(teacher_prefix.split())
```

In [59]:

```
#teacher_prefix_cat_dict = dict(my_counter)
#sorted_teacher_prefix_cat_dict = dict(sorted(teacher_prefix_cat_dict.items(), key=lambda kv: kv[1
]))
```

In [60]:

```
## we use count vectorizer to convert the values into one hot encoded features
## Unlike the previous Categories this category returns a
## ValueError: np.nan is an invalid document, expected byte or unicode string.
## The link below explains h0w to tackle such discrepancies.
## https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-
is-an-invalid-document/39308809#39308809

vectorizer_teacher = CountVectorizer(lowercase=False,binary=True)
vectorizer_teacher.fit(X_train['clean_teacher_prefix'].values.astype("U"))

teacher_prefix_categories_one_hot_train =
vectorizer_teacher.transform(X_train['clean_teacher_prefix'].values.astype("U"))
teacher_prefix_categories_one_hot_test =
vectorizer_teacher.transform(X_test['clean_teacher_prefix'].values.astype("U"))
#teacher_prefix_categories_one_hot_cv =
vectorizer_teacher.transform(X_cv['clean_teacher_prefix'].values.astype("U"))

print(vectorizer_teacher.get_feature_names())
#print(teacher_prefix_categories_one_hot_train[:,1:5])
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_test.shape)
#print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_cv.shape)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher', 'nan']
Shape of matrix after one hot encoding  (73196, 6)
Shape of matrix after one hot encoding  (36052, 6)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

Train Data - Essays

In [61]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).

vectorizer_bow_essay = CountVectorizer(min_df=10,max_features=5000,ngram_range=(1, 2))
vectorizer_bow_essay.fit(preprocessed_essays_train)

text_bow_train = vectorizer_bow_essay.transform(preprocessed_essays_train)

print("Shape of matrix after one hot encoding ",text_bow_train.shape)
```

```
Shape of matrix after one hot encoding  (73196, 5000)
```

Test Data - Essays

In [62]:

```
text_bow_test = vectorizer_bow_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 5000)
```

In [63]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

Train Data - Title

In [64]:

```
vectorizer_bow_title = CountVectorizer(min_df=10,max_features=5000,ngram_range=(1, 2))

vectorizer_bow_title.fit(preprocessed_titles_train)

title_bow_train = vectorizer_bow_title.transform(preprocessed_titles_train)
print("Shape of matrix after one hot encoding ",title_bow_train.shape)
```

```
Shape of matrix after one hot encoding  (73196, 4515)
```

Test Data - Title

In [65]:

```
title_bow_test = vectorizer_bow_title.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 4515)
```

1.5.2.2 TFIDF vectorizer

Train Data - Essays

In [66]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10,max_features=5000,ngram_range=(1, 2))
vectorizer_tfidf_essay.fit(preprocessed_essays_train)

text_tfidf_train = vectorizer_tfidf_essay.transform(preprocessed_essays_train)
print("Shape of matrix after one hot encoding ",text_tfidf_train.shape)
```

```
Shape of matrix after one hot encoding  (73196, 5000)
```

Test Data - Essays

In [67]:

```
text_tfidf_test = vectorizer_tfidf_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 5000)
```

Train Data - Title

In [68]:

```
vectorizer_tfidf_titles = TfidfVectorizer(min_df=10,max_features=5000,ngram_range=(1, 2))

vectorizer_tfidf_titles.fit(preprocessed_titles_train)
title_tfidf_train = vectorizer_tfidf_titles.transform(preprocessed_titles_train)
print("Shape of matrix after one hot encoding ",title_tfidf_train.shape)
```

```
Shape of matrix after one hot encoding  (73196, 4515)
```

Test Data - Title

In [69]:

```
title_tfidf_test = vectorizer_tfidf_titles.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 4515)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [70]:

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

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

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

```
# ============================

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

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

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

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


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

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


'''
```

Out[70]:

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

In [71]:

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

Train Data - Essays

In [72]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_train = [];

for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
```

```
for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 73196/73196
[00:34<00:00, 2120.50it/s]
```

```
73196
300
```

Test Data - Essays

In [73]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_test = [];

for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 36052/36052
[00:16<00:00, 2205.91it/s]
```

```
36052
300
```

Train Data - Title

In [74]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_train.append(vector)

print(len(avg_w2v_vectors_titles_train))
print(len(avg_w2v_vectors_titles_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████| 73196/73196
[00:01<00:00, 52511.43it/s]
```

```
73196
300
```

Test - Titles

In [75]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_test.append(vector)

print(len(avg_w2v_vectors_titles_test))
print(len(avg_w2v_vectors_titles_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████| 36052/36052
[00:00<00:00, 55150.05it/s]
```

```
36052
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

Train Data - Essays

In [76]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer(min_df=10,max_features=5000,ngram_range=(1, 2))
tfidf_model.fit(preprocessed_essays_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [77]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_train.append(vector)
```

```
print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████| 73196/73196 [04:02<00:00, 302.20it/s]
```

```
73196
300
```

Test Data - Essays

In [78]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████| 36052/36052 [01:55<00:00, 312.40it/s]
```

```
36052
300
```

In [79]:

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

Train Data - title

In [80]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer(min_df=10,max_features=5000,ngram_range=(1, 2))
tfidf_model.fit(preprocessed_titles_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [81]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
```

```
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_train.append(vector)

print(len(tfidf_w2v_vectors_titles_train))
print(len(tfidf_w2v_vectors_titles_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████| 73196/73196
[00:02<00:00, 33212.24it/s]
```

```
73196
300
```

Test Data - Title

In [82]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_test.append(vector)

print(len(tfidf_w2v_vectors_titles_test))
print(len(tfidf_w2v_vectors_titles_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████| 36052/36052
[00:01<00:00, 35587.31it/s]
```

```
36052
300
```

1.5.3 Vectorizing Numerical features

Price

In [83]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[83]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [86]:

```
# join two dataframes in python:
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
#X_cv = pd.merge(X_cv, price_data, on='id', how='left')
```

In [87]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

normalizer.fit(X_train['price'].values.reshape(1,-1))

price_train = normalizer.transform(X_train['price'].values.reshape(1,-1))
#price_cv = normalizer.transform(X_cv['price'].values.reshape(1,-1))
price_test = normalizer.transform(X_test['price'].values.reshape(1,-1))

print("After vectorizations")
print(price_train.shape, y_train.shape)
#print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
print("="*100)
print(price_train)
```

```
After vectorizations
(1, 73196) (73196,)
(1, 36052) (36052,)
====================================================================================================

[[3.08691103e-03 5.41408606e-04 2.26945985e-03 ... 3.86676224e-04
  1.43599388e-03 7.15637269e-05]]
```

Quantity

In [88]:

```
normalizer = Normalizer()

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

normalizer.fit(X_train['quantity'].values.reshape(1,-1))

quantity_train = normalizer.transform(X_train['quantity'].values.reshape(1,-1))
#quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(1,-1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(1,-1))

print("After vectorizations")
print(quantity_train.shape, y_train.shape)
#print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
print("="*100)
print(quantity_train)
```

```
After vectorizations
(1, 73196) (73196,)
(1, 36052) (36052,)
```

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

[[0.00023393 0.00140359 0.0037429  ... 0.00093572 0.00023393 0.00140359]]
```

Number of Projects previously proposed by Teacher

In [89]:

```
normalizer = Normalizer()

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

normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

prev_projects_train = normalizer.transform(X_train['teacher_number_of_previously_posted_projects']
.values.reshape(1,-1))
#prev_projects_cv =
normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].v
alues.reshape(1,-1))

print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
#print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
print("="*100)
print(prev_projects_train)
```

```
After vectorizations
(1, 73196) (73196,)
(1, 36052) (36052,)
==========================================================================================

[[0.00024714 0.         0.00259499 ... 0.000865   0.00024714 0.00135928]]
```

Title word Count

In [90]:

```
normalizer = Normalizer()

normalizer.fit(X_train['title_word_count'].values.reshape(1,-1))

title_word_count_train = normalizer.transform(X_train['title_word_count'].values.reshape(1,-1))
#title_word_count_cv = normalizer.transform(X_cv['title_word_count'].values.reshape(1,-1))
title_word_count_test = normalizer.transform(X_test['title_word_count'].values.reshape(1,-1))

print("After vectorizations")
print(title_word_count_train.shape, y_train.shape)
#print(title_word_count_cv.shape, y_cv.shape)
print(title_word_count_test.shape, y_test.shape)
print("="*100)
print(title_word_count_train)
```

```
After vectorizations
(1, 73196) (73196,)
(1, 36052) (36052,)
==========================================================================================

[[0.00132975 0.00465413 0.00199463 ... 0.00332438 0.00199463 0.00332438]]
```

Essay word Count

In [91]:

```
normalizer = Normalizer()

normalizer.fit(X_train['essay_word_count'].values.reshape(1,-1))

essay_word_count_train = normalizer.transform(X_train['essay_word_count'].values.reshape(1,-1))
#essay_word_count_cv = normalizer.transform(X_cv['essay_word_count'].values.reshape(1,-1))
essay_word_count_test = normalizer.transform(X_test['essay_word_count'].values.reshape(1,-1))

print("After vectorizations")
print(essay_word_count_train.shape, y_train.shape)
#print(essay_word_count_cv.shape, y_cv.shape)
print(essay_word_count_test.shape, y_test.shape)
print("="*100)
print(essay_word_count_train )
```

```
After vectorizations
(1, 73196) (73196,)
(1, 36052) (36052,)
====================================================================================================

[[0.00269347 0.00565348 0.00265138 ... 0.00338086 0.0030021  0.00266541]]
```

Assignment 4: 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

---

Logistic Regression

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (`BOW with bi-grams` with `min_df=10` and `max_features=5000`

In [92]:

```
price_train = (price_train.reshape(-1,1))
#price_cv = (price_cv.reshape(-1,1))
price_test = (price_test.reshape(-1,1))

quantity_train =(quantity_train.reshape(-1,1))
#quantity_cv = (quantity_cv.reshape(-1,1))
quantity_test = (quantity_test.reshape(-1,1))

prev_projects_train = (prev_projects_train.reshape(-1,1))
#prev_projects_cv = (prev_projects_cv.reshape(-1,1))
prev_projects_test = (prev_projects_test.reshape(-1,1))

title_word_count_train = (title_word_count_train.reshape(-1,1))
#title_word_count_cv = (title_word_count_cv.reshape(-1,1))
title_word_count_test = (title_word_count_test.reshape(-1,1))


essay_word_count_train = (essay_word_count_train.reshape(-1,1))
#essay_word_count_cv = (essay_word_count_cv.reshape(-1,1))
essay_word_count_test = (essay_word_count_test.reshape(-1,1))

print(price_train)
```

```
[[3.08691103e-03]
 [5.41408606e-04]
 [2.26945985e-03]
 ...
 [3.86676224e-04]
 [1.43599388e-03]
 [7.15637269e-05]]
```

In [93]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train, project_grade_categories_one_hot_train,
teacher_prefix_categories_one_hot_train, price_train, quantity_train, prev_projects_train, title_wo
rd_count_train, essay_word_count_train, title_bow_train, text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test, project_grade_categories_one_hot_test,
teacher_prefix_categories_one_hot_test, price_test, quantity_test, prev_projects_test,
title_word_count_test, essay_word_count_test, title_bow_test, text_bow_test)).tocsr()
#X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv, project_grade_categories_one_hot_cv,
teacher_prefix_categories_one_hot_cv, price_cv, quantity_cv, prev_projects_cv,
title_word_count_cv, essay_word_count_cv, title_bow_cv, text_bow_cv)).tocsr()
```

In [94]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
```

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

```
Final Data matrix
(73196, 9620) (73196,)
(36052, 9620) (36052,)
====================================================================================================
```

In [95]:

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

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

    return y_data_pred
```

In [96]:

```
def pred_prob(clf, data):
    y_pred = []
    y_pred = clf.predict_proba(data)[:,1]
    return y_pred
```

A) Random alpha values

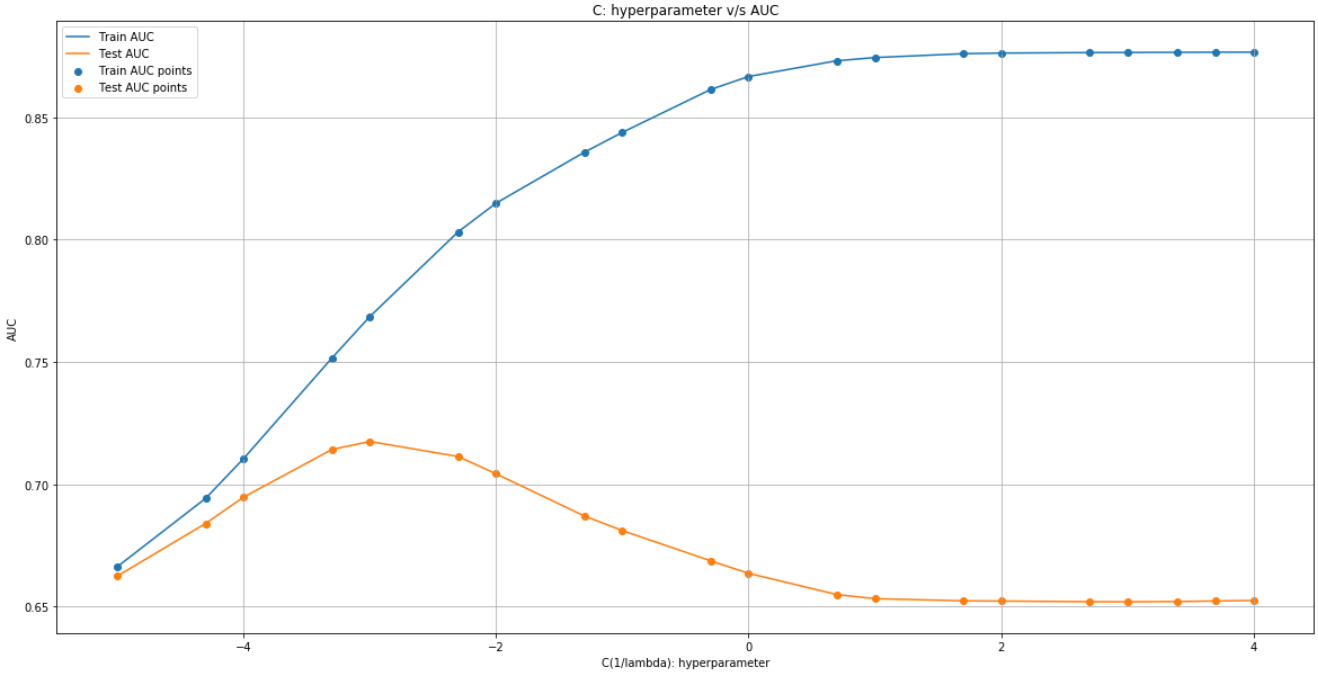
In [98]:

```
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import LogisticRegression

train_auc = []
test_auc = []
log_c =[]

Clist = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 50
0, 1000, 2500, 5000, 10000]

for i in tqdm(Clist):
    clf = LogisticRegression(C=i,class_weight='balanced');

    clf.fit(X_tr, y_train)
    y_train_pred = clf.predict_proba(X_tr)[:,1]
    y_test_pred = clf.predict_proba(X_te)[:,1]

    #y_train_pred = batch_predict(nb, X_tr)
    #y_cv_pred = batch_predict(nb, X_cr)


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    test_auc.append(roc_auc_score(y_test, y_test_pred))

for a in tqdm(Clist):
    b = np.log10(a)
    log_c.append(b)
```

```
100%|████████████████████████████████████████████████████████████████████████████| 20/20
[2:41:04<00:00, 1758.43s/it]
100%|████████████████████████████████████████████████████████████████████████████| 2
0/20 [00:00<?, ?it/s]
```

In [100]:

```
plt.figure(figsize=(20,10))

plt.plot(log_c, train_auc, label='Train AUC')
plt.plot(log_c, test_auc, label='Test AUC')

plt.scatter(log_c, train_auc, label='Train AUC points')
plt.scatter(log_c, test_auc, label='Test AUC points')

plt.legend()
plt.xlabel("C(1/lambda): hyperparameter")
plt.ylabel("AUC")
plt.title("C: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



B) Gridsearch-cv (K fold cross validation)

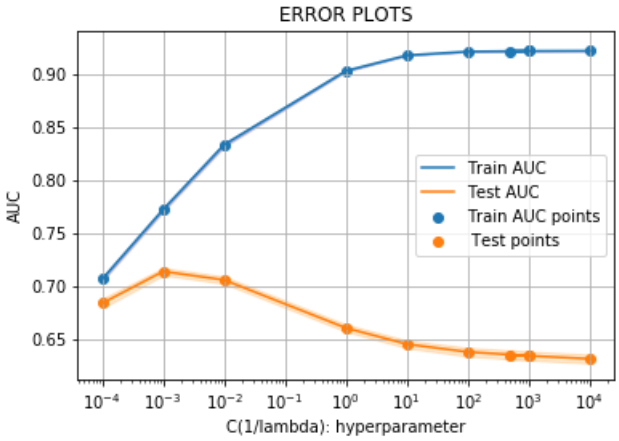
In [101]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(class_weight='balanced');
parameters ={'C':[10**-4, 10**-3,10**-2,1,10,100,1000,500,1000,10000]}
sd=GridSearchCV(clf, parameters, scoring='roc_auc',return_train_score=True)
sd.fit(X_tr, y_train);

train_auc= sd.cv_results_['mean_train_score']
train_auc_std= sd.cv_results_['std_train_score']
test_auc = sd.cv_results_['mean_test_score']
test_auc_std= sd.cv_results_['std_test_score']
```

In [103]:

```
import math
import matplotlib.pyplot as plt
```

```
plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['C'], test_auc, label='Test AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],test_auc - test_auc_std,test_auc + test_auc_std,alpha=0.2,co
lor='darkorange')

plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], test_auc, label=' Test points')
plt.xscale('log')

plt.legend()
plt.xlabel("C(1/lambda): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Summary

C values ranging from 0.0001 to 1000.0 was taken and the following results were obtained

1. Values closer to 0.001 works pretty well both on Train data and Cross Validation data.
2. Values more than 0.001 also doesnt seem to be effective on both Train and Cross Validation data.

C value is 0.001

C) Train model using the best hyper-parameter value

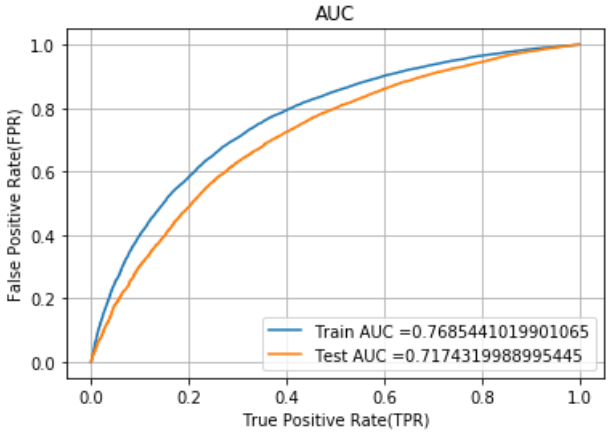
In [104]:

```
best_c = 10**-3
```

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=10**-3,class_weight='balanced');

neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = neigh.predict_proba(X_tr)[:,1]
```

```
y_test_pred = neigh.predict_proba(X_te)[:,1]
#y_train_pred = batch_predict(nb_bow, X_tr)
#y_test_pred = batch_predict(nb_bow, X_te)

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix

In [106]:

```
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(fpr*(1-tpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Train Data

In [107]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.412
[[ 5542  5541]
 [ 9120 52993]]
```

In [108]:

In [108]:

```
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
train_fpr, train_fpr)), range(2),range(2))
```

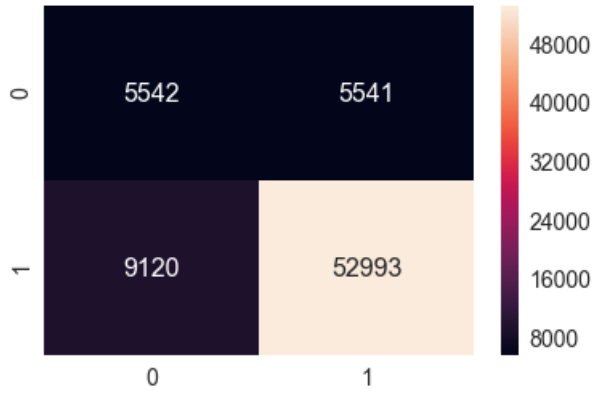
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.412
```

In [109]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f622f8e860>
```



Test Data

In [110]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.412
[[ 2346  3113]
 [ 4811 25782]]
```

In [111]:

```
conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, tra
in_fpr, train_fpr)), range(2),range(2))
```

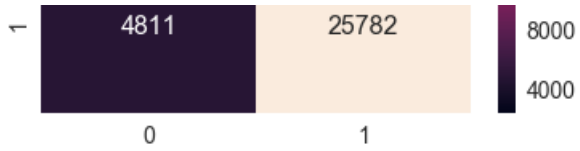
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.412
```

In [112]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[112]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f61f3d1e48>
```



Set 2 : categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (`TFIDF with bi-grams with min_df=10 and max_features=5000`)

In [113]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train, quantity_train,
               prev_projects_train, title_word_count_train, essay_word_count_train,
text_tfidf_train, title_tfidf_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test, quantity_test,
               prev_projects_test, title_word_count_test, essay_word_count_test, text_tfidf_test, t
itle_tfidf_test)).tocsr()
#X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
              # project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv
, quantity_cv,
               #prev_projects_cv, title_word_count_cv, essay_word_count_cv, text_tfidf_cv, title_t
idf_cv)).tocsr()
```

In [114]:

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

```
Final Data matrix
(73196, 9620) (73196,)
(36052, 9620) (36052,)
====================================================================================================
```

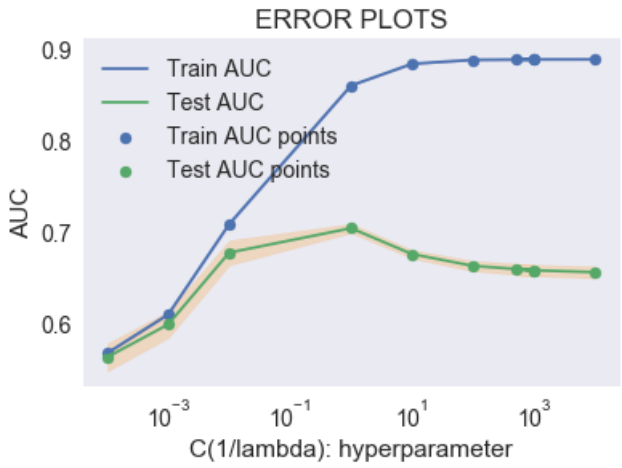
B) Gridsearch-cv using cv = 10 (K fold cross validation)

In [116]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(class_weight='balanced');
parameters ={'C':[10**-4, 10**-3,10**-2,1,10,100,1000,500,1000,10000]}
sd=GridSearchCV(clf, parameters,cv=10, scoring='roc_auc',return_train_score=True)
sd.fit(X_tr, y_train);

train_auc= sd.cv_results_['mean_train_score']
train_auc_std= sd.cv_results_['std_train_score']
test_auc = sd.cv_results_['mean_test_score']
test_auc_std= sd.cv_results_['std_test_score']
```

In [117]:

```
import math
import matplotlib.pyplot as plt



plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['C'], test_auc, label='Test AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],test_auc - test_auc_std,test_auc + test_auc_std,alpha=0.2,co
lor='darkorange')

plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], test_auc, label='Test AUC points')
plt.xscale('log')

plt.legend()
plt.xlabel("C(1/lambda): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Summary

C values ranging from 0.0001 to 1000.0 was taken and the following results were obtained :

1. 1000 as C values seemed to work very well on train data and the model seems to not work that efficiently on cross validation data.
2. Values closer to 0.1 and 1 works pretty well both on Train data and Cross Validation data.
3. Values more than 1 doesnt seem to be effective on both Train and Cross Validation data

C value chosen is 0.1

Train model using the best hyper-parameter value

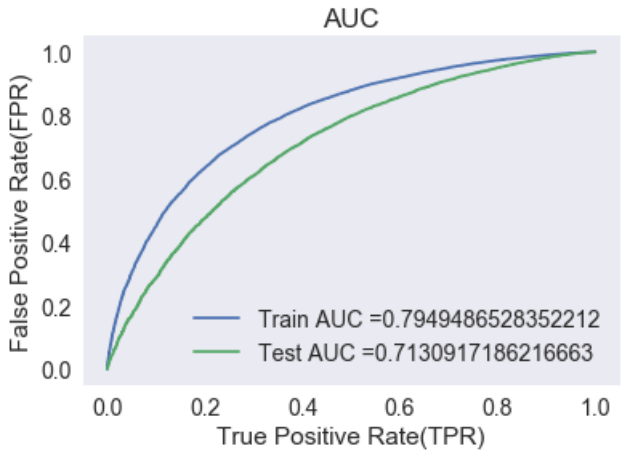
In [193]:

```
best_c2=0.1
```

In [119]:

```
# 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=best_c2,class_weight='balanced');

neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = neigh.predict_proba(X_tr)[:,1]
y_test_pred = neigh.predict_proba(X_te)[:,1]
#y_train_pred = batch_predict(nb_bow, X_tr)
#y_test_pred = batch_predict(nb_bow, X_te)

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix

Train Data

In [120]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.403
[[ 5542  5541]
 [ 7491 54622]]
```

In [121]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
train_fpr, train_fpr)), range(2),range(2))
```

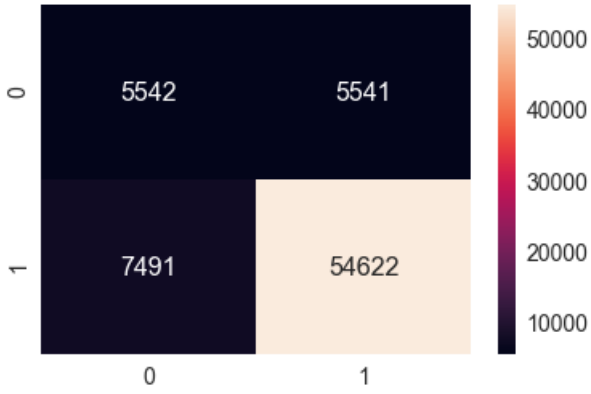
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.403
```

In [122]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[122]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f63c398240>
```



Test Data

In [123]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.403
[[ 2156  3303]
 [ 4215 26378]]
```

In [124]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, tra
in_fpr, train_fpr)), range(2),range(2))
```

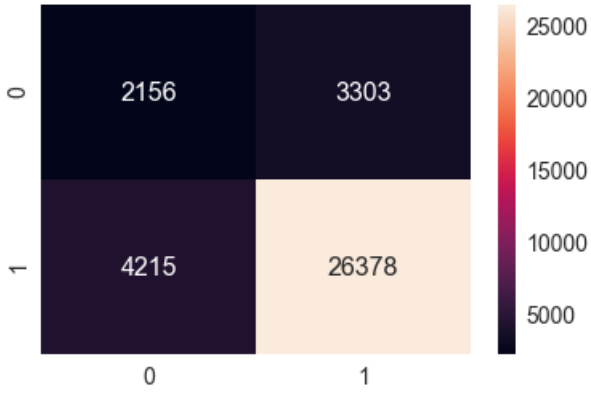
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.403
```

In [125]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[125]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f61cc89390>
```



SET 3 : AVG W2V

In [126]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train, quantity_train,
               prev_projects_train, title_word_count_train, essay_word_count_train,
avg_w2v_vectors_train, avg_w2v_vectors_titles_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test, quantity_test,
               prev_projects_test, title_word_count_test, essay_word_count_test,
avg_w2v_vectors_test, avg_w2v_vectors_titles_test)).tocsr()
#X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               #project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv
, quantity_cv,
               #prev_projects_cv, title_word_count_cv, essay_word_count_cv, avg_w2v_vectors_cv, avg
_w2v_vectors_titles_cv)).tocsr()
```

In [127]:

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

```
Final Data matrix
(73196, 705) (73196,)
(36052, 705) (36052,)
====================================================================================================
```

Select Random values

In [130]:

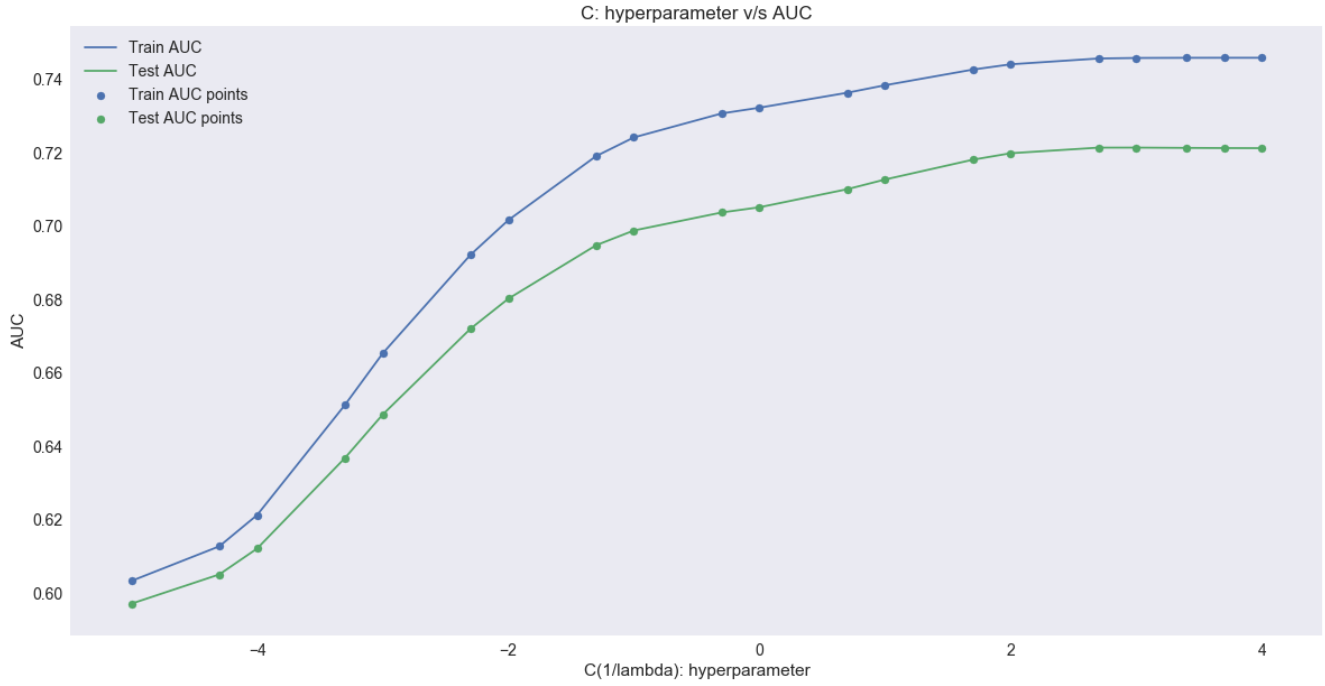
```
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import LogisticRegression

train_auc = []
test_auc = []
log_c =[]

Clist = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 50
0, 1000, 2500, 5000, 10000]

for i in tqdm(Clist):
    clf = LogisticRegression(C=i,class_weight='balanced');

    clf.fit(X_tr, y_train)
    y_train_pred = clf.predict_proba(X_tr)[:,1]
    y_test_pred = clf.predict_proba(X_te)[:,1]

    #y_train_pred = batch_predict(nb, X_tr)
    #y_cv_pred = batch_predict(nb, X_cr)


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    test_auc.append(roc_auc_score(y_test, y_test_pred))
```

```
for a in tqdm(Clist):
    b = np.log10(a)
    log_c.append(b)
```

```
100%|████████████████████████████████████████████████████████████████████████| 20/20
[1:22:47<00:00, 656.36s/it]
100%|████████████████████████████████████████████████████████████████████████| 2
0/20 [00:00<?, ?it/s]
```

In [131]:

```
plt.figure(figsize=(20,10))

plt.plot(log_c, train_auc, label='Train AUC')
plt.plot(log_c, test_auc, label='Test AUC')

plt.scatter(log_c, train_auc, label='Train AUC points')
plt.scatter(log_c, test_auc, label='Test AUC points')

plt.legend()
plt.xlabel("C(1/lambda): hyperparameter")
plt.ylabel("AUC")
plt.title("C: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



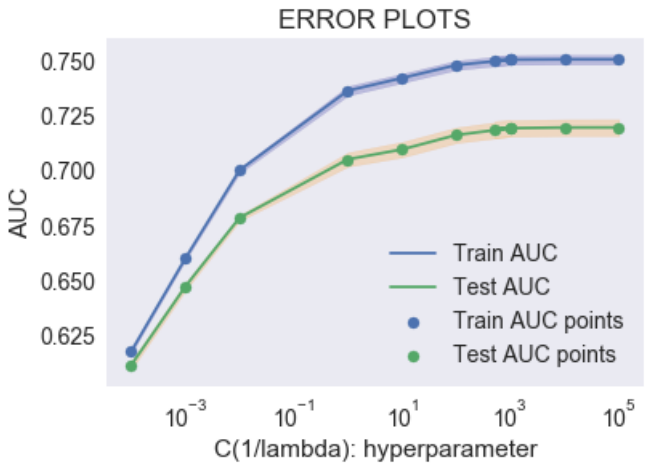
Grid Search

In [132]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(class_weight='balanced');
parameters ={'C':[10**-4, 10**-3,10**-2,1,10,100,1000,500,1000,10000,100000]}
sd=GridSearchCV(clf, parameters, scoring='roc_auc',return_train_score=True)
sd.fit(X_tr, y_train);

train_auc= sd.cv_results_['mean_train_score']
train_auc_std= sd.cv_results_['std_train_score']
test_auc = sd.cv_results_['mean_test_score']
test_auc_std= sd.cv_results_['std_test_score']
```

In [133]:

```
import math
import matplotlib.pyplot as plt



plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['C'], test_auc, label='Test AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],test_auc - test_auc_std,test_auc + test_auc_std,alpha=0.2,co
lor='darkorange')

plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], test_auc, label='Test AUC points')
plt.xscale('log')

plt.legend()
plt.xlabel("C(1/lambda): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Conclusion

C values ranging from 0.00001 to 100000.0 was taken and the following results were obtained

1. Values closer to 1000 works pretty well both on Train data and Cross Validation data.
2. Values less than 1000 also doesnt seem to be effective on both Train and Cross Validation data.

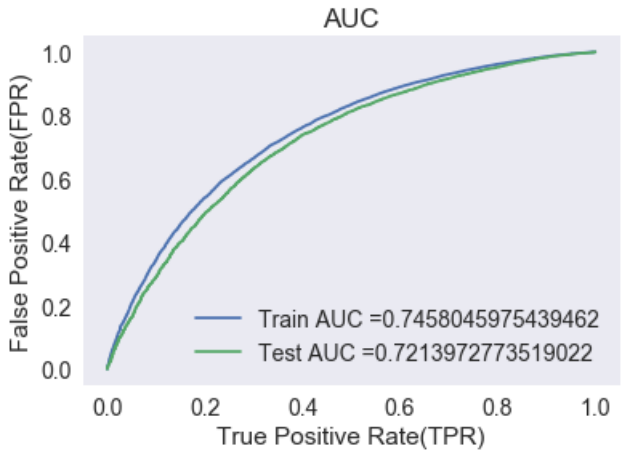
In [134]:

```
best_c3=10**3
```

In [135]:

```
# 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=best_c3,class_weight='balanced');

neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
```

```
y_train_pred = neigh.predict_proba(X_tr)[:,1]
y_test_pred = neigh.predict_proba(X_te)[:,1]
#y_train_pred = batch_predict(nb_bow, X_tr)
#y_test_pred = batch_predict(nb_bow, X_te)

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix

Train Data

In [136]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.39
[[ 5542  5541]
 [10315 51798]]
```

In [137]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
train_fpr, train_fpr)), range(2),range(2))
```

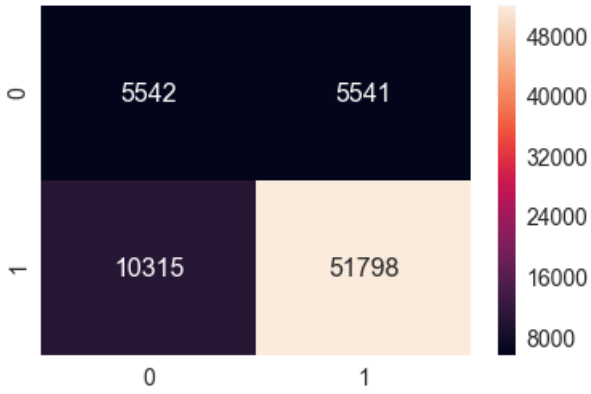
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.39
```

In [138]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[138]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f61f1e9198>
```



Test Data

In [139]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.39
[[ 1925  3534]
 [ 3249 27344]]
```

In [140]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, tra
in_fpr, train_fpr)), range(2),range(2))
```

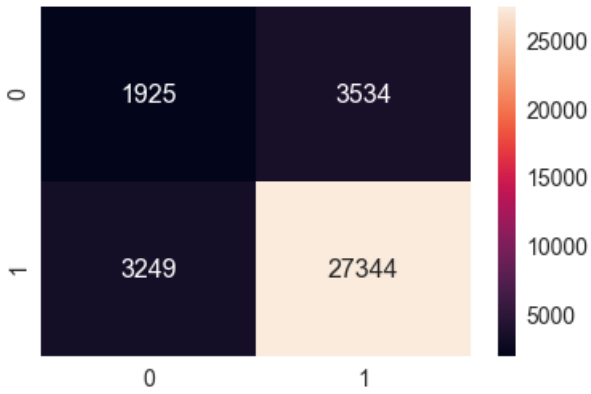
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.39
```

In [141]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[141]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f620be5780>
```



4. TF-IDF W2V

In [142]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
```

```
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train, quantity_train,
               prev_projects_train, title_word_count_train, essay_word_count_train,
tfidf_w2v_vectors_train, tfidf_w2v_vectors_titles_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test, quantity_test,
               prev_projects_test, title_word_count_test, essay_word_count_test,
tfidf_w2v_vectors_test, tfidf_w2v_vectors_titles_test)).tocsr()
#X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               #project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv
, quantity_cv,
               #prev_projects_cv, title_word_count_cv, essay_word_count_cv, avg_w2v_vectors_cv, avg
_w2v_vectors_titles_cv)).tocsr()
```

In [143]:

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

```
Final Data matrix
(73196, 705) (73196,)
(36052, 705) (36052,)
====================================================================================================
```

Grid Search CV

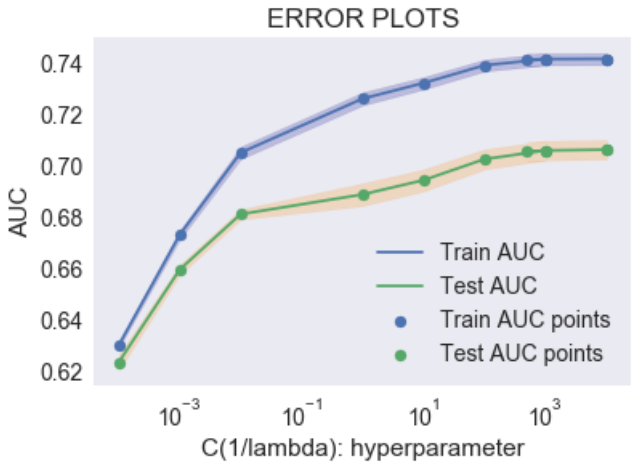
In [144]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(class_weight='balanced');
parameters ={'C':[10**-4, 10**-3,10**-2,1,10,100,1000,500,1000,10000,10000]}
sd=GridSearchCV(clf, parameters, scoring='roc_auc',return_train_score=True)
sd.fit(X_tr, y_train);

train_auc= sd.cv_results_['mean_train_score']
train_auc_std= sd.cv_results_['std_train_score']
test_auc = sd.cv_results_['mean_test_score']
test_auc_std= sd.cv_results_['std_test_score']
```

In [145]:

```
import math
import matplotlib.pyplot as plt



plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['C'], test_auc, label='Test AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],test_auc - test_auc_std,test_auc + test_auc_std,alpha=0.2,co
lor='darkorange')

plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], test_auc, label='Test AUC points')
plt.xscale('log')
```

```
plt.legend()
plt.xlabel("C(1/lambda): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Conclusion

C values ranging from 0.00001 to 10000.0 was taken and the following results were obtained

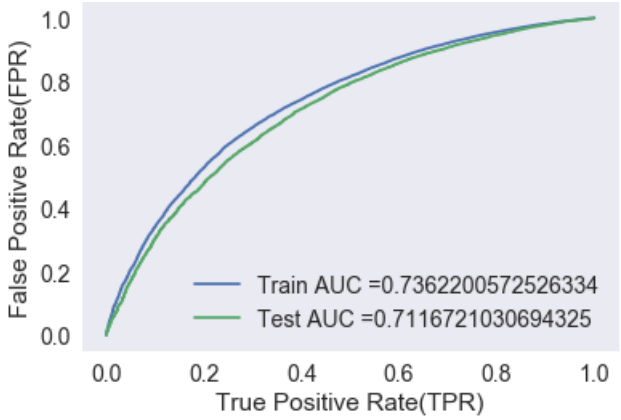
1. Values closer to 1000 works pretty well both on Train data and Cross Validation data.
2. Values less than 1000 also doesnt seem to be effective on both Train and Cross Validation data.

In [147]:

```
best_c4=10**3
```

In [148]:

```
# 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=best_c4,class_weight='balanced');

neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = neigh.predict_proba(X_tr)[:,1]
y_test_pred = neigh.predict_proba(X_te)[:,1]
#y_train_pred = batch_predict(nb_bow, X_tr)
#y_test_pred = batch_predict(nb_bow, X_te)

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix

Train Data

In [149]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
[[ 5542  5541]
 [11509 50604]]
```

In [150]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
train_fpr, train_fpr)), range(2),range(2))
```

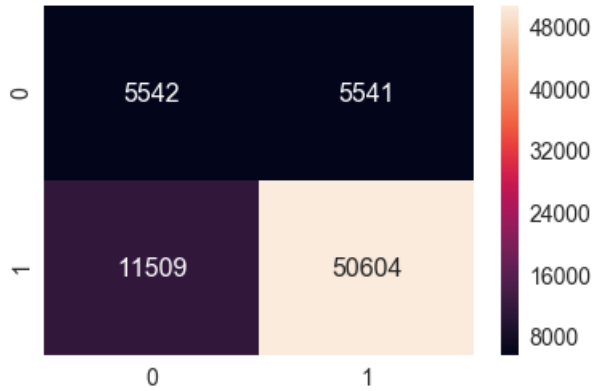
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
```

In [151]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[151]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6230d51d0>
```



Test Data

In [152]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
[[ 1908  3551]
 [ 3514 27079]]
```

In [153]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, tra
in_fpr, train_fpr)), range(2),range(2))
```

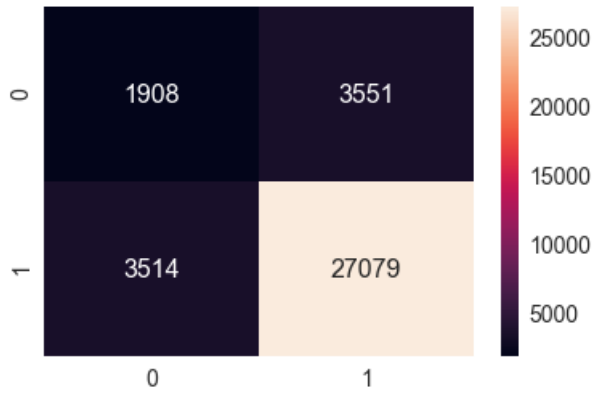
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
```

In [154]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[154]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f6235071d0>
```



New feature - Sentiment scores of each combined essay's

In [198]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')


#https://www.programcreek.com/python/example/100005/nltk.sentiment.vader.SentimentIntensityAnalyzer

def analyze_sentiment(df):
    sentiments = []
    sid = SentimentIntensityAnalyzer()
    for i in range(df.shape[0]):
        line = df['essay'][i]# take one essay
        sentiment = sid.polarity_scores(line)# calculate the sentiment
        sentiments.append([sentiment['neg'], sentiment['pos'],
                           sentiment['neu'], sentiment['compound']])# list of lists
    df[['neg', 'pos', 'neu', 'compound']] = pd.DataFrame(sentiments)
    df['Negative'] = df['compound'] < -0.1
    df['Positive'] = df['compound'] > 0.1
    return df
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\bkacharl\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [199]:

```
X_train=analyze_sentiment(X_train)
X_test=analyze_sentiment(X_test)
#X_cv=analyze_sentiment(X_cv)
```

In [200]:

```
#for train

pos=list(X_train['pos'])
pos=np.array(pos)
neg=list(X_train['neg'])
neg=np.array(neg)
com=list(X_train['compound'])
com=np.array(com)

# combine all
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_set5_train = hstack((
                       teacher_prefix_categories_one_hot_train,categories_one_hot_train,
sub_categories_one_hot_train, school_state_categories_one_hot_train,
                project_grade_categories_one_hot_train,#all categorials
                      price_train, quantity_train,
                prev_projects_train,
                       essay_word_count_train,title_word_count_train,
                       pos.reshape(-1,1),neg.reshape(-1,1),com.reshape(-1,1),
                                                                        ))# all numericals


print(X_set5_train.shape, y_train.shape)
```

```
(73196, 108) (73196,)
```

In [201]:

```
#for test

pos=list(X_test['pos'])
pos=np.array(pos)
neg=list(X_test['neg'])
neg=np.array(neg)
com=list(X_test['compound'])
com=np.array(com)

# combine all
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_set5_test = hstack((
                       teacher_prefix_categories_one_hot_test,categories_one_hot_test,
sub_categories_one_hot_test, school_state_categories_one_hot_test,
                project_grade_categories_one_hot_test,#all categorials
                      price_test, quantity_test,
                prev_projects_test,
                       essay_word_count_test,title_word_count_test,
                       pos.reshape(-1,1),neg.reshape(-1,1),com.reshape(-1,1),
                                                                        ))# all numericals


print(X_set5_test.shape, y_test.shape)
```

```
(36052, 108) (36052,)
```

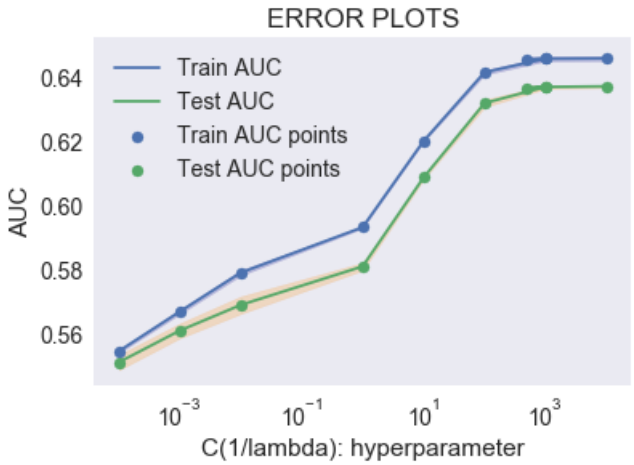
logistic regression on SET 5

In [202]:

```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
#from sklearn.grid_search import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import learning_curve, GridSearchCV

"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

clf = LogisticRegression(class_weight='balanced');
parameters ={'C':[10**-4, 10**-3,10**-2,1,10,100,1000,500,1000,10000]}
cl = GridSearchCV(clf, parameters, cv=3, scoring='roc_auc',return_train_score=True)
cl.fit(X_set5_train, y_train);

train_auc= cl.cv_results_['mean_train_score']
train_auc_std= cl.cv_results_['std_train_score']
test_auc = cl.cv_results_['mean_test_score']
test_auc_std= cl.cv_results_['std_test_score']
```

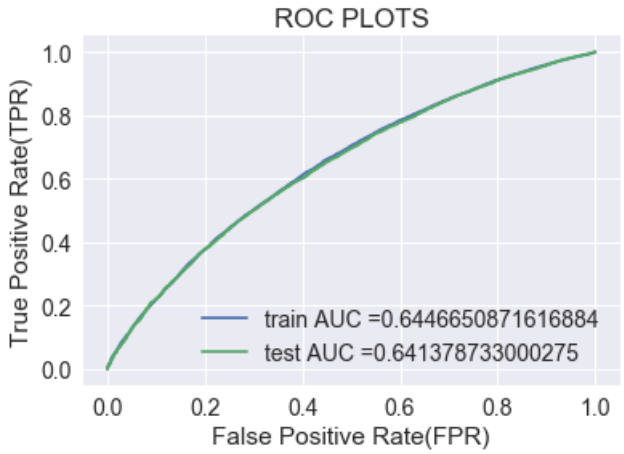
In [203]:

```
import math
import matplotlib.pyplot as plt

plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['C'], test_auc, label='Test AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],test_auc - test_auc_std,test_auc + test_auc_std,alpha=0.2,co
lor='darkorange')

plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], test_auc, label='Test AUC points')
plt.xscale('log')

plt.legend()
plt.xlabel("C(1/lambda): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [204]:

```
# 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**3,class_weight='balanced');
neigh.fit(X_set5_train ,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train, neigh.predict_proba(X_set5_train)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, neigh.predict_proba(X_set5_test)[:,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.ylabel("True Positive Rate(TPR)")
plt.xlabel("False Positive Rate(FPR)")
plt.title("ROC PLOTS")
plt.show()
```



Conculsion

C values ranging from 0.00001 to 10000.0 was taken and the following results were obtained

1. Values closer to 1000 works pretty well both on Train data and Cross Validation data.
2. Values less than 1000 also doesnt seem to be effective on both Train and Cross Validation data.

In [184]:

```
best_c5=10**3
```

Confusion matrix

Train Data

In [187]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

==========

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
[[ 5560  5523]
 [11559 50554]]
```

In [188]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

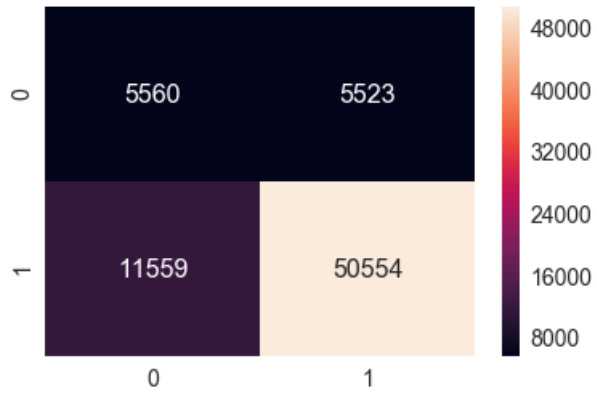
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
```

In [189]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[189]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f621bbb160>
```



Test Data

In [190]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_fpr)))
```

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

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
[[ 1911  3548]
 [ 3531 27062]]
```

In [191]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

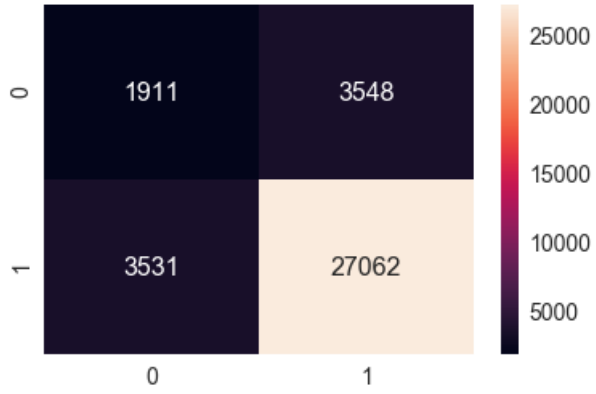
```
the maximum value of tpr*(1-fpr) 0.2499999979647145 for threshold 0.401
```

In [192]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[192]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f61f52a160>
```



Conclusions

In [205]:

```
# Please compare all your models using Prettytable library
# Please compare all your models using Prettytable library
#how to use pretty table http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

tb = PrettyTable()
tb.field_names= ("Vectorizer", "Model", "HyperParameter", "AUC")
tb.add_row(["BOW", "Auto",10**-3, 72])
tb.add_row(["Tf-Idf", "Auto",1, 71])
tb.add_row(["AVGW2V", "Auto",10**3, 72])
tb.add_row(["Tf-Idf w2v", "Auto", 10**3, 71])
tb.add_row(["Set 5", "Auto",10**3, 64])
print(tb.get_string(titles = "Logistic Reg> - Observations"))
```

```
+------------+-------+----------------+-----+
| Vectorizer | Model | HyperParameter | AUC |
+------------+-------+----------------+-----+
|    BOW     |  Auto |     0.001      |  72 |
|   Tf-Idf   |  Auto |       1        |  71 |
|   AVGW2V   |  Auto |      1000      |  72 |
| Tf-Idf w2v |  Auto |      1000      |  71 |
|   Set 5    |  Auto |      1000      |  64 |
+------------+-------+----------------+-----+
```