Document Created Date:08-03-2019

DonorsChoose.org
Support a classroom. Build a future.

Grasping the Data and DataSource - 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	
`project_id`	A unique identifier for the proposed project. **Exampl**
`project_title`	Title of the projec • `Art Will Make Y` • `First`

Feature	
`project_grade_category`	Grade level of students for which the project is targeted. One of t enumera • `Grad` • `G` • `G` • `Gr`
`project_subject_categories`	One or more (comma-separated) subject categories for the pro following enumerated li • `Applied` • `Care` • `Health` • `History` • `Literacy &` • `Math` • `Music &` • `Spec` • • `Music &` • `Literacy & Language, Math`
`school_state`	State where school is located ([Two-letter U.S. (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos) **Ex**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories fo • • `Literature & Writing, Social`
`project_resource_summary`	An explanation of the resources needed for the projec • `My students need hands on literacy materials t sensory nee`
`project_essay_1`	First applic
`project_essay_2`	Second applic
`project_essay_3`	Third applic
`project_essay_4`	Fourth applic
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2( 12:`
`teacher_id`	A unique identifier for the teacher of the proposed proje `bdf8baa8fedef6bfeec7ae4`
`teacher_prefix`	Teacher's title. One of the following enumera • • • • • •
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the sa **I**

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

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

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

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

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

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

Notes on the Essay Data

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

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_3:** "Close by sharing why your project will make a difference"

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

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

```
Step by Step Procedure
```

- Understanding the Businessreal world problem
- Loading the data
- Preprocessing the data(based on the type of data = categorical , text, Numarical)
- Preprocessing data includes (removing outliers, impute missung values, cleaning data, remove spacial character, etc..)
- Split the data into train, cv, test(random splitting)
- Vectorization data (one hot encoding)
- Vectorizing text data(bow, tfidf, avgw2v, tfidf weighted w2v)
- vectorizing numarical - Normalizer

- Computing Sentiment Scores
- Applying Logistic Regression
- Contactinating all the type of features(cat + text + num)
- Hyper parameter Tuning to find alpha:: Simple cross Validation (applied two techniques - this is one)
- Hyperparameter tuning to find th best estimator(RandomizedSearchCV- 2nd technique)
- Train the Logistic Regression model using best hyperparameter and ploting auc roc-curve
- Ploting confusion matrix(heatmaps)
- Observation on overall model performences
- Ploting the performences by tableu format.

```
C:\Users\Ramesh Battu> import required libraries
```

In [3]:

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

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [4]:

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

In [5]:

```
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 [6]:

```
# 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.col


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
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[6]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	0(
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	0(

In [7]:

```
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[7]:

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

1.1.1 preprocessing of `project_subject_categories`

In [8]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47

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

1.1.2 preprocessing of project_subject_subcategories

In [9]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47

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

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

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

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

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

1.1.3 preprocessing of school_state

In [10]:

```
school_state_catogories = list(project_data['school_state'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47

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

project_data['school_state'] = cat_list


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

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

1.1.4 preprocessing of teacher_prefix

In [11]:

```
# citation code :https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['teacher_prefix'].value_counts().index[
teacher_prefix_catogories = list(project_data['teacher_prefix'].values)
# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-sciki
# To convert the data type object to unicode string : used """astype('U')""" code from
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in teacher_prefix_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['teacher_prefix'] = cat_list


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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv: kv
```

1.1.5 Preprocessing of project_grade_category

In [12]:

```
# Feature encoding with 'project_grade_category'
project_grade_catogories = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in project_grade_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "Ma
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Ma
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['project_grade_category'] = cat_list

#link : https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['project_grade_category'].value_counts(

# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-sciki
# To convert the data type object to unicode string : used """astype('U')""" code from
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039

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



# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grade_category_dict = dict(my_counter)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), ke
```

1.2. Text Preprocessing

1.2.1 Text Preprocessing of essay

In [13]:

```
# 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 [14]:

```
project_data.head(1)
```

Out[14]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00

In [15]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[170])
print("="*50)
print(project_data['essay'].values[1989])
print("="*50)
print(project_data['essay'].values[30000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my class
room as well as the STEM journals, which my students really enjoyed.  I
would love to implement more of the Lakeshore STEM kits in my classroom
for the next school year as they provide excellent and engaging STEM les
sons.My students come from a variety of backgrounds, including language
and socioeconomic status.  Many of them don't have a lot of experience i
n science and engineering and these kits give me the materials to provid
e these exciting opportunities for my students.Each month I try to do se
veral science or STEM/STEAM projects.  I would use the kits and robot to
help guide my science instruction in engaging and meaningful ways.  I ca
n adapt the kits to my current language arts pacing guide where we alrea
dy teach some of the material in the kits like tall tales (Paul Bunyan)
or Johnny Appleseed.  The following units will be taught in the next sch
ool year where I will implement these kits: magnets, motion, sink vs. fl
oat, robots.  I often get to these units and don't know If I am teaching
the right way or using the right materials.    The kits will give me add
itional ideas, strategies, and lessons to prepare my students in scienc
e.It is challenging to develop high quality science activities.  These k
its give me the materials I need to provide my students with science act
```

In [16]:

```
# 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 [17]:

```
sent = decontracted(project_data['essay'].values[3000])
print(sent)
print("="*125)
```

```
\"Any book that helps a child to form a habit of reading, to make reading
one of his deep and continuing needs, is good for him.\" -Richard McKenna.
We live in an area where students do not always take pride in their educat
ion, and I am trying to change that through reading.CCMS is a Title 1 scho
ol, where a lot of students do not value their education. I hear on a dail
y basis, \"I hate reading!\", \"It is all boring!\". A lot of the students
live in poverty. They do not come to school with pencils, paper, or even b
ackpacks, let alone a book they are reading for pleasure. I want to bring
out the inner reader in all of my students.It is expected of my students t
o read several books a year. However, with little funds in a Title 1 schoo
l, it is hard for me to constantly add new and exciting books to my classr
oom library when I am having to spend my money on pencils, paper, notebook
s, etc. I want to inspire students to discover their abilities in reading
which will help them reach their own potential. I have learned the most im
portant thing is to have all students engaged in the lessons and reading.
In order to have your students engaged, you must know your students. I  By
taking interest in each of my children as an individual and what they like
to read, I will be able to bring their cultural identities into the readin
g. With a more interesting and up to date classroom library, my children w
ill be able to learn and share the same love of reading I have.In order to
turn my students into lifelong readers, I need to have more current books
available for them to read. Instead of teaching these books I am asking fo
r, I will be teaching comprehension strategies and literary elements that
students can then apply to these books we will be adding to the classroom!
Giving my students the independence to pick their own books will strengthe
n their self-confidence, and promote a positive attitude toward reading.
============================================================================
=================================================
```

In [18]:

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

```
 Any book that helps a child to form a habit of reading, to make reading o
ne of his deep and continuing needs, is good for him.  -Richard McKenna. W
e live in an area where students do not always take pride in their educati
on, and I am trying to change that through reading.CCMS is a Title 1 schoo
l, where a lot of students do not value their education. I hear on a daily
basis,  I hate reading  ,  It is all boring  . A lot of the students live
in poverty. They do not come to school with pencils, paper, or even backpa
cks, let alone a book they are reading for pleasure. I want to bring out t
he inner reader in all of my students.It is expected of my students to rea
d several books a year. However, with little funds in a Title 1 school, it
is hard for me to constantly add new and exciting books to my classroom li
brary when I am having to spend my money on pencils, paper, notebooks, et
c. I want to inspire students to discover their abilities in reading which
will help them reach their own potential. I have learned the most importan
t thing is to have all students engaged in the lessons and reading. In ord
er to have your students engaged, you must know your students. I  By takin
g interest in each of my children as an individual and what they like to r
ead, I will be able to bring their cultural identities into the reading. W
ith a more interesting and up to date classroom library, my children will
be able to learn and share the same love of reading I have.In order to tur
n my students into lifelong readers, I need to have more current books ava
ilable for them to read. Instead of teaching these books I am asking for,
I will be teaching comprehension strategies and literary elements that stu
dents can then apply to these books we will be adding to the classroom  Gi
ving my students the independence to pick their own books will strengthen
their self-confidence, and promote a positive attitude toward reading.
```

In [19]:

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

```
 Any book that helps a child to form a habit of reading to make reading on
e of his deep and continuing needs is good for him Richard McKenna We live
in an area where students do not always take pride in their education and
I am trying to change that through reading CCMS is a Title 1 school where
a lot of students do not value their education I hear on a daily basis I h
ate reading It is all boring A lot of the students live in poverty They do
not come to school with pencils paper or even backpacks let alone a book t
hey are reading for pleasure I want to bring out the inner reader in all o
f my students It is expected of my students to read several books a year H
owever with little funds in a Title 1 school it is hard for me to constant
ly add new and exciting books to my classroom library when I am having to
spend my money on pencils paper notebooks etc I want to inspire students t
o discover their abilities in reading which will help them reach their own
potential I have learned the most important thing is to have all students
engaged in the lessons and reading In order to have your students engaged
you must know your students I By taking interest in each of my children as
an individual and what they like to read I will be able to bring their cul
tural identities into the reading With a more interesting and up to date c
lassroom library my children will be able to learn and share the same love
of reading I have In order to turn my students into lifelong readers I nee
d to have more current books available for them to read Instead of teachin
g these books I am asking for I will be teaching comprehension strategies
and literary elements that students can then apply to these books we will
be adding to the classroom Giving my students the independence to pick the
ir own books will strengthen their self confidence and promote a positive
attitude toward reading
```

In [20]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
            "as","at","be","because","been","before","being","below","between","both",
            "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
            "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
            "here","hers","herself","him","himself","his","how","i","if","in","into","
            "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
            "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
            "out","over","own","re","s","same","shan","shan't","she","she's","should",
            "so","some","such","t","than","that","that'll","the","their","theirs","the
            "these","they","this","those","through","to","too","under","until","up","v
            "we","were","weren","weren't","what","when","where","which","while","who",
            "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
            "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
            "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
            "what's","when's","where's","who's","why's","would","able","abst","accorda
            "across","act","actually","added","adj","affected","affecting","affects","
            "along","already","also","although","always","among","amongst","announce",
            "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
            "around","aside","ask","asking","auth","available","away","awfully","b","b
            "becoming","beforehand","begin","beginning","beginnings","begins","behind"
            "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
            "co","com","come","comes","contain","containing","contains","couldnt","dat
            "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsew
            "especially","et","etc","even","ever","every","everybody","everyone","ever
            "f","far","ff","fifth","first","five","fix","followed","following","follow
            "found","four","furthermore","g","gave","get","gets","getting","give","giv
            "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
            "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
            "importance","important","inc","indeed","index","information","instead","i
            "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
            "later","latter","latterly","least","less","lest","let","lets","like","lik
            "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
            "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
            "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
            "neither","never","nevertheless","new","next","nine","ninety","nobody","no
            "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
            "omitted","one","ones","onto","ord","others","otherwise","outside","overal
            "particular","particularly","past","per","perhaps","placed","please","plus
            "potentially","pp","predominantly","present","previously","primarily","pro
            "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
            "recently","ref","refs","regarding","regardless","regards","related","rela
            "resulted","resulting","results","right","run","said","saw","say","saying"
            "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
            "shes","show","showed","shown","showns","shows","significant","significant
            "six","slightly","somebody","somehow","someone","somethan","something","so
            "somewhere","soon","sorry","specifically","specified","specify","specifyin
            "sub","substantially","successfully","sufficiently","suggest","sup","sure"
            "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
            "therein","there'll","thereof","therere","theres","thereto","thereupon","t
            "thou","though","thoughh","thousand","throug","throughout","thru","thus","
            "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
            "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
            "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
            "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
            "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
            "who'll","whomever","whos","whose","widely","willing","wish","within","wit
            "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
            "appreciate","appropriate","associated","best","better","c'mon","c's","can
            "consequently","consider","considering","corresponding","course","currentl
            "entirely","exactly","example","going","greetings","hello","help","hopeful
            "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","p
            "secondly","sensible","serious","seriously","sure","t's","third","thorough
            "wonder"]
```

In [21]:

```
%time
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = sent.replace('!', ' ')
    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.append(sent.lower().strip())
```

```
Wall time: 0 ns

100%|██████████████████████████████████| 109248/109248 [06:13<00:00, 292.76
it/s]
```

In [22]:

```
# after preprocesing
preprocessed_essays[30000]
```

Out[22]:

```
'school title school student receives free breakfast lunch parents student
s children succeed limited happen child families live areas parents comfor
table children playing sit front tv video games minimal physical activity
children play fun children active inside classroom confined rigid plastic
chair day school reasons medical physical students constant motion sitting
chair concentrating classwork difficult sitting floor chair desk feet chai
r squatting chair standing work class ordinary normal kids inspired projec
t students asked seating required sit conventional student seating working
classwork physically sit factors pillows cushions standing work students a
ssignment find seats concentrate learn wanted seating find write project s
tudents internet looked classrooms types seating hokki stools love colors
orange orange favorite color told main reason work sit learning movement e
xcising determined mentally physically fit time sitting stools 60 minutes
day 60 minutes day exercise'
```

1.2.2 Text Preprocessing of project_title

In [23]:

```
print(project_data['project_title'].tail(1))
```

```
78306    News for Kids
Name: project_title, dtype: object
```

In [24]:

```
# printing some random title texts
print(project_data['project_title'].values[100])
print('--'*19)
print(project_data['project_title'].values[129])
print('--'*19)
print(project_data['project_title'].values[1120])
print('--'*19)
print(project_data['project_title'].values[99999])
print('--'*19)
```

```
iCan with iPads...and YOU!
--------------------------------------
Teaching with Technology
--------------------------------------
Trojan Pride Tumbling and Stunting
--------------------------------------
Turning to Flexible Seating: One Sixth-Grade Class's Journey to Freedom
--------------------------------------
```

In [25]:

```
# 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 [26]:

```
sent = decontracted(project_data['project_title'].values[99999])
print(sent)
print("="*125)
```

```
Turning to Flexible Seating: One Sixth-Grade Class is Journey to Freedom
==========================================================================
===================================================
```

In [27]:

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

```
Turning to Flexible Seating: One Sixth-Grade Class is Journey to Freedom
```

In [28]:

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

```
Turning to Flexible Seating One Sixth Grade Class is Journey to Freedom
```

In [29]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
            "as","at","be","because","been","before","being","below","between","both",
            "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
            "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
            "here","hers","herself","him","himself","his","how","i","if","in","into","
            "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
            "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
            "out","over","own","re","s","same","shan","shan't","she","she's","should",
            "so","some","such","t","than","that","that'll","the","their","theirs","the
            "these","they","this","those","through","to","too","under","until","up","v
            "we","were","weren","weren't","what","when","where","which","while","who",
            "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
            "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
            "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
            "what's","when's","where's","who's","why's","would","able","abst","accorda
            "across","act","actually","added","adj","affected","affecting","affects","
            "along","already","also","although","always","among","amongst","announce",
            "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
            "around","aside","ask","asking","auth","available","away","awfully","b","b
            "becoming","beforehand","begin","beginning","beginnings","begins","behind"
            "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
            "co","com","come","comes","contain","containing","contains","couldnt","dat
            "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsew
            "especially","et","etc","even","ever","every","everybody","everyone","ever
            "f","far","ff","fifth","first","five","fix","followed","following","follow
            "found","four","furthermore","g","gave","get","gets","getting","give","giv
            "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
            "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
            "importance","important","inc","indeed","index","information","instead","i
            "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
            "later","latter","latterly","least","less","lest","let","lets","like","lik
            "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
            "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
            "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
            "neither","never","nevertheless","new","next","nine","ninety","nobody","no
            "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
            "omitted","one","ones","onto","ord","others","otherwise","outside","overal
            "particular","particularly","past","per","perhaps","placed","please","plus
            "potentially","pp","predominantly","present","previously","primarily","pro
            "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
            "recently","ref","refs","regarding","regardless","regards","related","rela
            "resulted","resulting","results","right","run","said","saw","say","saying"
            "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
            "shes","show","showed","shown","showns","shows","significant","significant
            "six","slightly","somebody","somehow","someone","somethan","something","so
            "somewhere","soon","sorry","specifically","specified","specify","specifyin
            "sub","substantially","successfully","sufficiently","suggest","sup","sure"
            "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
            "therein","there'll","thereof","therere","theres","thereto","thereupon","t
            "thou","though","thoughh","thousand","throug","throughout","thru","thus","
            "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
            "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
            "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
            "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
            "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
            "who'll","whomever","whos","whose","widely","willing","wish","within","wit
            "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
```

```
                "appreciate","appropriate","associated","best","better","c'mon","c's","can
                "consequently","consider","considering","corresponding","course","currentl
                "entirely","exactly","example","going","greetings","hello","help","hopeful
                "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","p
                "secondly","sensible","serious","seriously","sure","t's","third","thorough
                "wonder"]
```

In [30]:

```
%time
# Combining all the above stundents
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = sent.replace('!', ' ')
    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_project_title.append(sent.lower().strip())
```

```
Wall time: 0 ns

100%|██████████████████████████████████| 109248/109248 [00:12<00:00, 8656.24
it/s]
```

In [31]:

```
preprocessed_project_title[99991]
```

Out[31]:

```
'media literacy project students severe profound disabilities'
```

1.3. Numerical normalization

1.3.1 normalization_price

In [32]:

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

Out[32]:

```
(109248, 20)
```

In [33]:

```
project_data.head(1)
```

Out[33]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	I
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2(04 00:2

In [34]:

```
print(project_data["price"].shape)
```

```
(109248,)
```

In [35]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(-1, 1)
price_normalize = Normalizer()
price_normalizer = price_normalize.fit_transform(project_data['price'].values.reshape(1
price_normalizer = price_normalizer.T
print(price_normalizer)
print("-------------------------------------------------------")
print("shape of price_normalizer:", price_normalizer.shape)
```

```
[[4.63560392e-03]
 [1.36200635e-03]
 [2.10346002e-03]
 ...
 [2.55100471e-03]
 [1.83960046e-03]
 [3.51642253e-05]]
-------------------------------------------------------
shape of price_normalizer: (109248, 1)
```

1.3.2 Normalization of teacher_number_of_previously_posted_projects

In [36]:

```
project_data["teacher_number_of_previously_posted_projects"].values
```

Out[36]:

```
array([53,  4, 10, ...,  0,  1,  2], dtype=int64)
```

In [37]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
teacher_number_of_previously_posted_projects_normalize = Normalizer()
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
print(teacher_number_of_previously_posted_projects_normalizer)
print("="*25)
print("Shape of teacher_number_of_previously_posted_projects_normalizer :", teacher_num
```

```
[[0.00535705]
 [0.00040431]
 [0.00101076]
 ...
 [0.        ]
 [0.00010108]
 [0.00020215]]
=========================
Shape of teacher_number_of_previously_posted_projects_normalizer : (10924
8, 1)
```

1.3.3 spilt the data into train ,CV and test

In [38]:

```
project_data.head(1)
```

Out[38]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Dat
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016 04-2 00:27:3

In [39]:

```
project_data['project_is_approved'].values
```

Out[39]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [40]:

```
# class label
label = project_data['project_is_approved']
project_data.drop(['project_is_approved'], axis=1, inplace=True)
```

In [41]:

```
# spliting the data into train , test CV
# Refrence link :https://scikit-learn.org/stable/modules/generated/sklearn.model_select
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data, label, test_size=0.33
X_train, X_cv,  y_train, y_cv    = train_test_split(X_train, y_train, test_size=0.33) #

print("Shape of X_train and y_train  :", X_train.shape, y_train.shape)
print("Shape of X_test and y_test    :", X_test.shape, y_test.shape)
print("Shape of X_cv and y_cv        :", X_cv.shape, y_cv.shape)
```

```
Shape of X_train and y_train  : (49041, 19) (49041,)
Shape of X_test and y_test    : (36052, 19) (36052,)
Shape of X_cv and y_cv        : (24155, 19) (24155,)
```

1.4. Vectorizing Categorical data

1.4.1 Vectorization of project_subject_categories¶

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

In [42]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, 

clean_categories_one_hot_train = vectorizer.fit_transform(X_train['clean_categories'].v
clean_categories_one_hot_test  = vectorizer.transform(X_test['clean_categories'].values
clean_categories_one_hot_cv    = vectorizer.transform(X_cv['clean_categories'].values)

print("vectorizer feature names :", vectorizer.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",clean_categories_one_hot_train.s
print("Shape of matrix after one hot encodig test  : ",clean_categories_one_hot_test.sh
print("Shape of matrix after one hot encodig cv    : ",clean_categories_one_hot_cv.shap
```

```
vectorizer feature names : ['Warmth', 'Care_Hunger', 'History_Civics', 'Mu
sic_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Scien
ce', 'Literacy_Language']
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 9)
Shape of matrix after one hot encodig test  :  (36052, 9)
Shape of matrix after one hot encodig cv    :  (24155, 9)
```

1.4.2 Vectorization of project_subject_subcategories

In [43]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal

clean_subcategories_one_hot_train = vectorizer.fit_transform(X_train['clean_subcategori
clean_subcategories_one_hot_test  = vectorizer.transform(X_test['clean_subcategories'].
clean_subcategories_one_hot_cv    = vectorizer.transform(X_cv['clean_subcategories'].va

print("vectorizer feature names :", vectorizer.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",clean_subcategories_one_hot_trai
print("Shape of matrix after one hot encodig test  : ",clean_subcategories_one_hot_test
print("Shape of matrix after one hot encodig cv    : ",clean_subcategories_one_hot_cv.s
```

```
vectorizer feature names : ['Economics', 'CommunityService', 'FinancialLit
eracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'Fore
ignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialScien
ces', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'Coll
ege_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'Earl
yDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts',
'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writin
g', 'Mathematics', 'Literacy']
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 30)
Shape of matrix after one hot encodig test  :  (36052, 30)
Shape of matrix after one hot encodig cv    :  (24155, 30)
```

1.4.3 Vectorization of school_state

In [44]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lowercas

school_state_one_hot_train = vectorizer.fit_transform(X_train['school_state'].values)
school_state_one_hot_test  = vectorizer.transform(X_test['school_state'].values)
school_state_one_hot_cv    = vectorizer.transform(X_cv['school_state'].values)

print(vectorizer.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",school_state_one_hot_train.shape
print("Shape of matrix after one hot encodig test  : ",school_state_one_hot_test.shape)
print("Shape of matrix after one hot encodig cv    : ",school_state_one_hot_cv.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME',
'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'N
V', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'M
A', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'N
Y', 'TX', 'CA']
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 51)
Shape of matrix after one hot encodig test  :  (36052, 51)
Shape of matrix after one hot encodig cv    :  (24155, 51)
```

1.4.4 Vectorization of teacher_prefix

In [45]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), lowerc


teacher_prefix_one_hot_train = vectorizer.fit_transform(X_train['teacher_prefix'].value
teacher_prefix_one_hot_test  = vectorizer.transform(X_test['teacher_prefix'].values.ast
teacher_prefix_one_hot_cv    = vectorizer.transform(X_cv['teacher_prefix'].values.astyp

print(vectorizer.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",teacher_prefix_one_hot_train.sha
print("Shape of matrix after one hot encodig test  : ",teacher_prefix_one_hot_test.shap
print("Shape of matrix after one hot encodig cv    : ",teacher_prefix_one_hot_cv.shape)
```

```
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 5)
Shape of matrix after one hot encodig test  :  (36052, 5)
Shape of matrix after one hot encodig cv    :  (24155, 5)
```

In [46]:

```
vectorizer.get_feature_names
```

Out[46]:

```
<bound method CountVectorizer.get_feature_names of CountVectorizer(analyze
r='word', binary=True, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=False, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None,
        vocabulary=['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.'])>
```

1.4.5 Vectorization of project_grade_category

In [47]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys())

project_grade_category_one_hot_train = vectorizer.fit_transform(X_train['project_grade_
project_grade_category_one_hot_test  = vectorizer.transform(X_test['project_grade_categ
project_grade_category_one_hot_cv    = vectorizer.transform(X_cv['project_grade_categor

print(vectorizer.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train  : ",project_grade_category_one_hot_
print("Shape of matrix after one hot encodig test   : ",project_grade_category_one_hot_
print("Shape of matrix after one hot encodig cv     : ",project_grade_category_one_hot_
```

```
['Grades9-12', 'Grades6-8', 'Grades3-5', 'GradesPreK-2']
---------------------------------------------------------
Shape of matrix after one hot encodig train  :  (49041, 4)
Shape of matrix after one hot encodig test   :  (36052, 4)
Shape of matrix after one hot encodig cv     :  (24155, 4)
```

1.5. Vectorizing Text

1.5.1 Vectorization of essays bow

In [48]:

```
X_train['essay'].tail(1)
```

Out[48]:

```
18375    We are a third grade language arts and social ...
Name: essay, dtype: object
```

In [49]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_essays_bow = CountVectorizer(preprocessed_essays, min_df=10, max_features=50

essays_bow_train = vectorizer_essays_bow.fit_transform(X_train['essay'].values)
essays_bow_test  = vectorizer_essays_bow.transform(X_test['essay'].values)
essays_bow_cv    = vectorizer_essays_bow.transform(X_cv['essay'].values)

print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",essays_bow_train.shape)
print("Shape of matrix after one hot encodig test  : ",essays_bow_test.shape)
print("Shape of matrix after one hot encodig cv    : ",essays_bow_cv.shape)
```

```
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 5000)
Shape of matrix after one hot encodig test  :  (36052, 5000)
Shape of matrix after one hot encodig cv    :  (24155, 5000)
```

In [50]:

```
vectorizer_essays_bow.get_feature_names
```

Out[50]:

```
<bound method CountVectorizer.get_feature_names of CountVectorizer(analyze
r='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8',
        input=['fortunate fairy tale stem kits classroom stem journals stu
dents enjoyed love implement lakeshore stem kits classroom school year pro
vide excellent engaging stem lessons students variety backgrounds includin
g language socioeconomic status lot experience science engineering kits ma
terials prov... standards interesting materials lead engaging discussions
inspire students find additional topics'],
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)>
```

1.5.1.1 Vectorization of essays tfidf

In [51]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_essays_tfidf = TfidfVectorizer(preprocessed_essays, min_df=10, max_features=

essays_tfidf_train = vectorizer_essays_tfidf.fit_transform(X_train['essay'].values)
essays_tfidf_test  = vectorizer_essays_tfidf.transform(X_test['essay'].values)
essays_tfidf_cv    = vectorizer_essays_tfidf.transform(X_cv['essay'].values)

print("Shape of matrix after one hot encodig of train : ",essays_tfidf_train.shape)
print("Shape of matrix after one hot encodig test     : ",essays_tfidf_test.shape)
print("Shape of matrix after one hot encodig cv       : ",essays_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig of train :  (49041, 5000)
Shape of matrix after one hot encodig test     :  (36052, 5000)
Shape of matrix after one hot encodig cv       :  (24155, 5000)
```

1.5.1.2 Using Pretrained Models: essays Avg W2V

In [52]:

```
'''
# 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_essays:
    words.extend(i.split(' '))

for i in preprocessed_project_title:
    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-p

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


'''
```

Out[52]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGlo
veModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(glove
File,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n
splitLine = line.split()\n        word = splitLine[0]\n        embedding =
np.array([float(val) for val in splitLine[1:]])\n        model[word] = emb
edding\n    print ("Done.",len(model)," words loaded!")\n    return model
\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ====================
========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/
s]\nDone. 1917495  words loaded!\n\n# ============================\n\nword
s = []\nfor i in preproced_essays:\n    words.extend(i.split(\' \'))\n\nfo
r i in preprocessed_project_title:\n    words.extend(i.split(\' \'))\nprin
t("all the words in the coupus", len(words))\nwords = set(words)\nprint("t
he unique words in the coupus", len(words))\n\ninter_words = set(model.key
s()).intersection(words)\nprint("The number of words that are present in b
oth 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        word
s_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n
\n\n# stronging variables into pickle files python: http://www.jessicayun
g.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport (h
ttp://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-
python/\n\nimport) pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n
pickle.dump(words_courpus, f)\n\n\n'
```

In [53]:

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

In [54]:

```
# average Word2Vec X_train
# compute average word2vec for each review.
essays_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(X_train['essay']): # 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
    essays_avg_w2v_vectors_train.append(vector)

print(len(essays_avg_w2v_vectors_train))
print(len(essays_avg_w2v_vectors_train[0]))
```

```
100%|████████████████████████████████████| 49041/49041 [00:49<00:00, 994.55
it/s]

49041
300
```

Average Word2Vec X_test_essay

In [55]:

```
# average Word2Vec X_test_essay
# compute average word2vec for each review.
essays_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in t
for sentence in tqdm(X_test['essay']): # 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
    essays_avg_w2v_vectors_test.append(vector)

print(len(essays_avg_w2v_vectors_test))
print(len(essays_avg_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████| 36052/36052 [00:36<00:00, 995.39
it/s]

36052
300
```

Average Word2Vec X_cv_essay

In [56]:

```
# average Word2Vec X_cv
# compute average word2vec for each review.
essays_avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in thi
for sentence in tqdm(X_cv['essay']): # 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
    essays_avg_w2v_vectors_cv.append(vector)

print(len(essays_avg_w2v_vectors_cv))
print(len(essays_avg_w2v_vectors_cv[0]))
```

```
100%|████████████████████████████████████| 24155/24155 [00:24<00:00, 992.63
it/s]

24155
300
```

1.5.1.3 essays TFIDF weighted W2V train

In [57]:

```
tfidf_model_preprocessed_essays_train = TfidfVectorizer()
tfidf_model_preprocessed_essays_train.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_essays_train.get_feature_names(), list(t
tfidf_words = set(tfidf_model_preprocessed_essays_train.get_feature_names())
```

In [58]:

```
# essays TFIDF weighted W2V_train
# compute average word2vec for each review.
preprocessed_essays_train_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/revie
for sentence in tqdm(X_train['essay']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_train_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_train_tfidf_w2v_vectors))
print(len(preprocessed_essays_train_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████| 49041/49041 [08:23<00:00, 97.34
it/s]

49041
300
```

essays TFIDF weighted W2V test

In [59]:

```
# tfidf_model_preprocessed_essays_test
tfidf_model_preprocessed_essays_test = TfidfVectorizer()
tfidf_model_preprocessed_essays_test.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_essays_test.get_feature_names(), list(tf
tfidf_words = set(tfidf_model_preprocessed_essays_test.get_feature_names())
```

In [60]:

```
# tfidf_model_preprocessed_essays_test
# compute average word2vec for each review.
preprocessed_essays_test_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review
for sentence in tqdm(X_test['essay']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_test_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_test_tfidf_w2v_vectors))
print(len(preprocessed_essays_test_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████| 36052/36052 [06:20<00:00, 94.67
it/s]

36052
300
```

essays TFIDF weighted W2V cv

In [61]:

```
# tfidf_model_preprocessed_essays_cv
tfidf_model_preprocessed_essays_cv = TfidfVectorizer()
tfidf_model_preprocessed_essays_cv.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_essays_cv.get_feature_names(), list(tfid
tfidf_words = set(tfidf_model_preprocessed_essays_cv.get_feature_names())
```

In [62]:

```
# tfidf_model_preprocessed_essays_cv
# compute average word2vec for each review.
preprocessed_essays_cv_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review i
for sentence in tqdm(X_cv['essay']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_cv_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_cv_tfidf_w2v_vectors))
print(len(preprocessed_essays_cv_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████| 24155/24155 [04:18<00:00, 93.55
it/s]

24155
300
```

1.5.2 Vectorization of project_title bow train, test, cv

In [63]:

```
X_train['project_title'].head(1)
```

Out[63]:

```
69717    Motivating Immigrant Teens in the Trump Era
Name: project_title, dtype: object
```

In [64]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pro
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_project_title_bow = CountVectorizer(preprocessed_project_title, min_df=10, ma

project_title_bow_train = vectorizer_project_title_bow.fit_transform(X_train['project_t
project_title_bow_test  = vectorizer_project_title_bow.transform(X_test['project_title'
project_title_bow_cv    = vectorizer_project_title_bow.transform(X_cv['project_title'].

print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",project_title_bow_train.shape)
print("Shape of matrix after one hot encodig test  : ",project_title_bow_test.shape)
print("Shape of matrix after one hot encodig cv    : ",project_title_bow_cv.shape)
```

```
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 4718)
Shape of matrix after one hot encodig test  :  (36052, 4718)
Shape of matrix after one hot encodig cv    :  (24155, 4718)
```

1.5.2.1 Vectorization of project_title tfidf train, test. cv

In [65]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pro
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_project_title_tfidf = TfidfVectorizer(preprocessed_project_title, min_df=10,

project_title_tfidf_train = vectorizer_project_title_tfidf.fit_transform(X_train['proje
project_title_tfidf_test  = vectorizer_project_title_tfidf.transform(X_test['project_ti
project_title_tfidf_cv    = vectorizer_project_title_tfidf.transform(X_cv['project_titl

print("Shape of matrix after one hot encodig of train : ",project_title_tfidf_train.sha
print("Shape of matrix after one hot encodig test     : ",project_title_tfidf_test.shap
print("Shape of matrix after one hot encodig cv       : ",project_title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig of train :  (49041, 4718)
Shape of matrix after one hot encodig test     :  (36052, 4718)
Shape of matrix after one hot encodig cv       :  (24155, 4718)
```

1.5.2.2 Using Pretrained Models: project_title Avg W2V train

In [66]:

```
'''
# 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_essays:
    words.extend(i.split(' '))

for i in preprocessed_project_title:
    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-p

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


'''
```

Out[66]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/382303
49/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) load
GloveModel(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        mod
el[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n
return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ======
=====================\nOutput:\n    \nLoading Glove Model\n1917495it [0
6:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# =================
==========\n\nwords = []\nfor i in preproced_essays:\n    words.extend
(i.split(\' \'))\n\nfor i in preprocessed_project_title:\n    words.exte
nd(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwo
rds = set(words)\nprint("the unique words in the coupus", len(words))\n
\ninter_words = set(model.keys()).intersection(words)\nprint("The number
of words that are present in both glove vectors and our coupus",       l
en(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]\nprint("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 (http://www.jessicayung.com/how-to-u
se-pickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith
 open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus,
```

In [67]:

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

In [68]:

```
# average Word2Vec  project_title_train
# compute average word2vec for each review.
project_title_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is sto
for sentence in tqdm(X_train['project_title']): # 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
    project_title_avg_w2v_vectors_train.append(vector)

print(len(project_title_avg_w2v_vectors_train))
print(len(project_title_avg_w2v_vectors_train[0]))
```

```
100%|██████████████████████████████████| 49041/49041 [00:00<00:00, 52223.86
it/s]

49041
300
```

Average Word2Vec project_title_test

In [70]:

```
# average Word2Vec  project_title_test
# compute average word2vec for each review.
project_title_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stor
for sentence in tqdm(X_test['project_title']): # 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
    project_title_avg_w2v_vectors_test.append(vector)

print(len(project_title_avg_w2v_vectors_test))
print(len(project_title_avg_w2v_vectors_test[0]))
```

```
100%|██████████████████████████████████| 36052/36052 [00:00<00:00, 49047.53
it/s]

36052
300
```

Average Word2Vec project_title_cv

In [71]:

```
# average Word2Vec  project_title_ cv
# compute average word2vec for each review.
project_title_avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_cv['project_title']): # 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
    project_title_avg_w2v_vectors_cv.append(vector)

print(len(project_title_avg_w2v_vectors_cv))
print(len(project_title_avg_w2v_vectors_cv[0]))
```

```
100%|██████████████████████████████████| 24155/24155 [00:00<00:00, 42901.64
it/s]

24155
300
```

1.5.2.3 project_title TFIDF weighted W2V train

In [72]:

```
tfidf_model_preprocessed_project_title_train = TfidfVectorizer()
tfidf_model_preprocessed_project_title_train.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_project_title_train.get_feature_names(),
tfidf_words = set(tfidf_model_preprocessed_project_title_train.get_feature_names())
```

In [73]:

```
# project_title TFIDF weighted W2V train
# compute average word2vec for each review.
preprocessed_project_title_train_tfidf_w2v_vectors = []; # the avg-w2v for each sentenc
for sentence in tqdm(X_train['project_title']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_train_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_train_tfidf_w2v_vectors))
print(len(preprocessed_project_title_train_tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████| 49041/49041 [00:01<00:00, 28965.27
it/s]

49041
300
```

project_title TFIDF weighted W2V_ test

In [74]:

```
# tfidf_model_preprocessed_project_title_test
tfidf_model_preprocessed_project_title_test = TfidfVectorizer()
tfidf_model_preprocessed_project_title_test.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_project_title_test.get_feature_names(),
tfidf_words = set(tfidf_model_preprocessed_project_title_test.get_feature_names())
```

In [75]:

```
# project_title TFIDF weighted W2V_ test
# compute average word2vec for each review.
preprocessed_project_title_test_tfidf_w2v_vectors = []; # the avg-w2v for each sentence
for sentence in tqdm(X_test['project_title']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_test_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_test_tfidf_w2v_vectors))
print(len(preprocessed_project_title_test_tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████| 36052/36052 [00:01<00:00, 28452.99
it/s]

36052
300
```

project_title TFIDF weighted W2V_cv

In [85]:

```
# tfidf_model_preprocessed_project_title_cv
tfidf_model_preprocessed_project_title_cv = TfidfVectorizer()
tfidf_model_preprocessed_project_title_cv.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_project_title_cv.get_feature_names(), li
tfidf_words = set(tfidf_model_preprocessed_project_title_cv.get_feature_names())
```

In [86]:

```
# project_title TFIDF weighted W2V_cv
# compute average word2vec for each review.
preprocessed_project_title_cv_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/r
for sentence in tqdm(X_cv['project_title']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_cv_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_cv_tfidf_w2v_vectors))
print(len(preprocessed_project_title_cv_tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████| 24155/24155 [00:04<00:00, 5478.25
it/s]

24155
300
```

1.6. Vectorizing Numerical features

1.6.1 Normalization of price_train_test_cv

In [161]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(-1, 1)
price_normalizer = Normalizer()
price_normalizer_train = price_normalizer.fit_transform(X_train['price'].values.reshape
price_normalizer_test  = price_normalizer.transform(X_test['price'].values.reshape(1,-1
price_normalizer_cv    = price_normalizer.transform(X_cv['price'].values.reshape(1,-1))

# https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html
# Transpose the array
price_normalizer_train = price_normalizer_train.T
price_normalizer_test  = price_normalizer_test.T
price_normalizer_cv    = price_normalizer_cv.T


print("shape of price_normalizer_train:", price_normalizer_train.shape)
print("---------------------------")
print(price_normalizer_train)

print("shape of price_normalizer_test :", price_normalizer_test.shape)
print("---------------------------")
print(price_normalizer_test)

print("shape of price_normalizer_cv   :", price_normalizer_cv.shape)
print("---------------------------")
print(price_normalizer_cv)
```

```
shape of price_normalizer_train: (49041, 1)
---------------------------
[[0.00132736]
 [0.0009633 ]
 [0.00416393]
 ...
 [0.00184522]
 [0.00178053]
 [0.00134053]]
shape of price_normalizer_test : (36052, 1)
---------------------------
[[0.00269115]
 [0.00387963]
 [0.00098527]
 ...
 [0.00383187]
 [0.00227344]
 [0.00478679]]
shape of price_normalizer_cv   : (24155, 1)
---------------------------
[[1.43285642e-03]
 [1.32563967e-03]
 [5.01506719e-04]
 ...
 [1.22650948e-03]
 [6.88752625e-05]
 [6.85755017e-03]]
```

1.6.2 Teacher_number_of_previously_posted_projects_train_test_cv : Numerical / Normalization

In [160]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(-1, 1)
teacher_number_of_previously_posted_projects_normalizer = Normalizer()

teacher_number_of_previously_posted_projects_normalizer_train = teacher_number_of_previ

teacher_number_of_previously_posted_projects_normalizer_test  = teacher_number_of_previ

teacher_number_of_previously_posted_projects_normalizer_cv    = teacher_number_of_previ


teacher_number_of_previously_posted_projects_normalizer_train=teacher_number_of_previou
teacher_number_of_previously_posted_projects_normalizer_test=teacher_number_of_previous
teacher_number_of_previously_posted_projects_normalizer_cv = teacher_number_of_previous



print("shape of teacher_number_of_previously_posted_projects_normalizer_train:",teacher_
print("----------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_train)

print("shape of teacher_number_of_previously_posted_projects_normalizer_test :",teacher_
print("----------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_test)

print("shape of teacher_number_of_previously_posted_projects_normalizer_cv   :",teacher_
print("----------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_cv)
```

```
shape of teacher_number_of_previously_posted_projects_normalizer_train:
(49041, 1)
----------------------------
[[0.00444395]
 [0.00122592]
 [0.        ]
 ...
 [0.        ]
 [0.0096541 ]
 [0.        ]]
shape of teacher_number_of_previously_posted_projects_normalizer_test :
(36052, 1)
----------------------------
[[0.00017838]
 [0.00017838]
 [0.00035677]
 ...
 [0.00035677]
 [0.00017838]
 [0.00856239]]
shape of teacher_number_of_previously_posted_projects_normalizer_cv    :
(24155, 1)
----------------------------
[[0.01269009]
 [0.00020468]
 [0.00020468]
 ...
 [0.        ]
 [0.00347954]
 [0.        ]]
```

In [ ]:

```
project_data.columns
```

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
```

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

1.7 Computing Sentiment Scores essay

In [171]:

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

# link :https://stackoverflow.com/questions/53680690/list-object-has-no-attribute-encod

vader = SentimentIntensityAnalyzer()
essay_train_str = X_train['essay'].str[0].str.join(" ")
sid = essay_train_str.apply(lambda x: vader.polarity_scores(x)['compound'])

# Normalization

normalizer = Normalizer()
sid_train = normalizer.fit_transform(sid.values.reshape(-1,1))
print(sid_train)

print("Shape of sid_train and y_train :", sid_train.shape, y_train.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sid_train and y_train : (49041, 1) (49041,)
```

In [172]:

```
# sid test

vader = SentimentIntensityAnalyzer()
essay_test_str = X_test['essay'].str[0].str.join(" ")
sid = essay_test_str.apply(lambda x: vader.polarity_scores(x)['compound'])

# Normalization

normalizer = Normalizer()
sid_test = normalizer.fit_transform(sid.values.reshape(-1,1))
print(sid_test)

print("Shape of sid_test and y_test :", sid_test.shape, y_test.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sid_test and y_test : (36052, 1) (36052,)
```

In [173]:

```
# sid cv

vader = SentimentIntensityAnalyzer()
essay_cv_str = X_cv['essay'].str[0].str.join(" ")
sid = essay_cv_str.apply(lambda x: vader.polarity_scores(x)['compound'])

# Normalization

normalizer = Normalizer()
sid_cv = normalizer.fit_transform(sid.values.reshape(-1,1))
print(sid_cv)

print("Shape of sid_cv and y_cv :", sid_cv.shape, y_cv.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sid_cv and y_cv : (24155, 1) (24155,)
```

Assignment : Logistic Regression

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

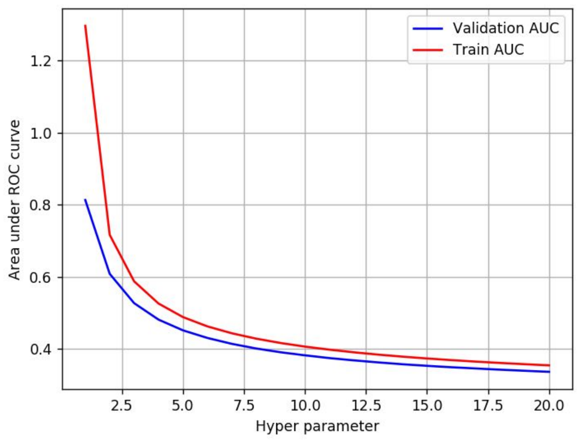
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (`BOW with bi-grams` with `min_df=10` and `max_features=5000`)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (`TFIDF with bi-grams` with `min_df=10` and `max_features=5000`)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

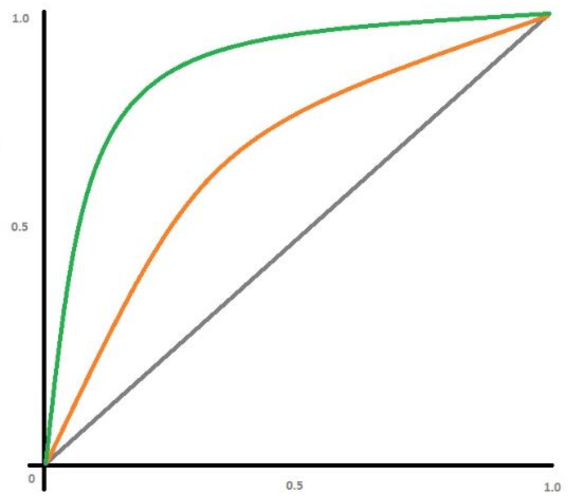
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - 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**

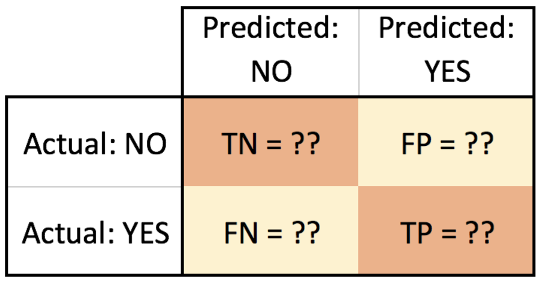
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure.



- 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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using [seaborn heatmaps. (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html)



	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

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 :

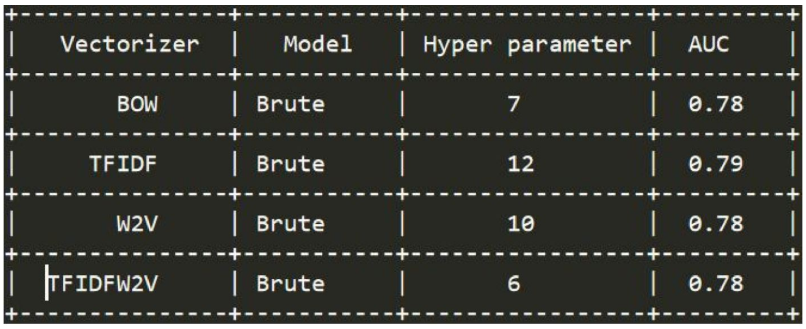
- **school_state** : categorical data
- **clean_categories** : categorical data
- **clean_subcategories** : categorical data
- **project_grade_category** :categorical data
- **teacher_prefix** : categorical data
- **quantity** : numerical data
- **teacher_number_of_previously_posted_projects** : numerical 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 (http://zetcode.com/python/prettytable/)](http://zetcode.com/python/prettytable/)



```
+----------------+-----------+------------------+---------+
|   Vectorizer   |   Model   | Hyper parameter  |   AUC   |
+----------------+-----------+------------------+---------+
|      BOW       |  Brute    |        7         |  0.78   |
+----------------+-----------+------------------+---------+
|     TFIDF      |  Brute    |        12        |  0.79   |
+----------------+-----------+------------------+---------+
|      W2V       |  Brute    |        10        |  0.78   |
+----------------+-----------+------------------+---------+
|   TFIDFW2V     |  Brute    |        6         |  0.78   |
+----------------+-----------+------------------+---------+
```

```
<h4><font color='red'>Note: Data Leakage</font></h4>

1. There will be an issue of data-leakage if you vectorize the entire data and then
split it into train/cv/test.
2. To avoid the issue of data-leakage, make sure to split your data first and then
vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data,
and apply the method transform() on cv/test data.
4. For more details please go through this <a href='https://soundcloud.com/applied-ai-
course/leakage-bow-and-tfidf'>link.</a>
```

2. Logistic Regression

2.1 Applying Logistic Regression on BOW, SET 1

Merging features encoding numerical + categorical features BOW, SET 1

In [82]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set1_ = all categorical features + numarical features + project_title(BOW) + preproce

set1_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     essays_bow_train,teacher_number_of_previously_posted_projects_norm
                     price_normalizer_train)).tocsr()


set1_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                    teacher_prefix_one_hot_test,project_grade_category_one_hot_test, pr
                    essays_bow_test,teacher_number_of_previously_posted_projects_normal
                    price_normalizer_test)).tocsr()

set1_cv  = hstack((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv,school_s
                   teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv,project_
                   essays_bow_cv,teacher_number_of_previously_posted_projects_normalize
                   price_normalizer_cv)).tocsr()

print("Final Data Matrix of set1 :")
print("shape of set1_train and y_train :", set1_train.shape , y_train.shape)
print("shape of set1_test and y_test   :", set1_test.shape , y_test.shape)
print("shape of set1_cv and y_cv       :", set1_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set1 :
shape of set1_train and y_train : (49041, 9804) (49041,)
shape of set1_test and y_test   : (36052, 9804) (36052,)
shape of set1_cv and y_cv       : (24155, 9804) (24155,)
```

2.1.1.A Hyper parameter Tuning to find alpha:: Simple cross Validation set1

In [100]:

```
%%time
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
params = [0.000001]
for i in params:
    # instantiate learning model (C = 10000)
    clf = LogisticRegression(C=i, class_weight='balanced',penalty='l2')
    # fitting the model on crossvalidation train

    clf.fit(set1_train, y_train)
    # predict the response on the crossvalidation train

    pred = clf.predict(set1_cv)
    # evaluate CV accuracy

    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)
    print('\nCV accuracy for lamda = %d is %d%%' % (i, acc))
```

```
CV accuracy for lamda = 0 is 84%
CPU times: user 1.15 s, sys: 8 ms, total: 1.16 s
Wall time: 1.16 s
```

In [98]:

```
clf.get_params
```

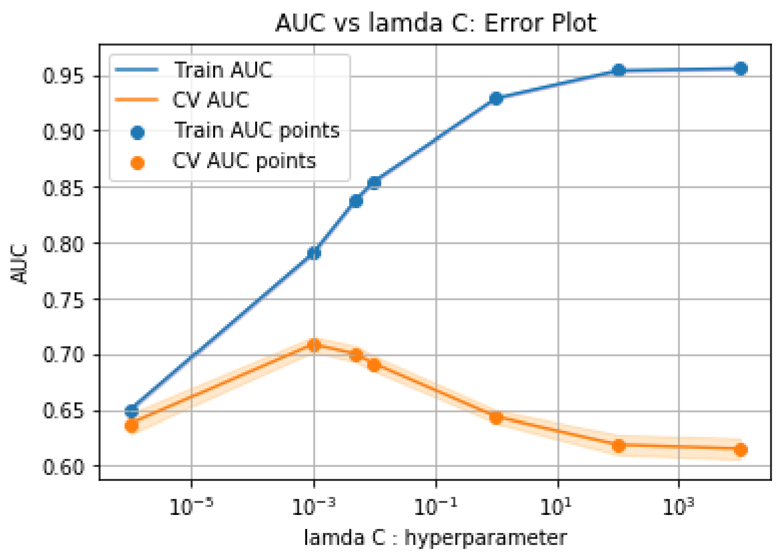
Out[98]:

```
<bound method BaseEstimator.get_params of LogisticRegression(C=1e-06, clas
s_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='warn', n_jobs=None, penalty
='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=
0,
                   warm_start=False)>
```

2.1.1.B Hyper parameter Tuning to find lamda(C) :: RandomizedSearchCV

In [84]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
LR = LogisticRegression(class_weight='balanced')
C = [0.000001, 0.001, 0.005, 0.01, 1, 10, 50]
params = {'C':[0.000001, 0.001, 0.005, 0.01, 1, 10, 50,]}
clf = RandomizedSearchCV(LR, params, cv=5,scoring='roc_auc', return_train_score=True)
clf.fit(set1_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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

plt.plot(params['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['C'],cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(params['C'], train_auc, label='Train AUC points')
plt.scatter(params['C'], cv_auc, label='CV AUC points')

plt.legend()
plt.xscale('log')
plt.xlabel("lamda C : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs lamda C: Error Plot")
plt.grid()
plt.show()
```



```
CPU times: user 2h 11min 2s, sys: 6.15 s, total: 2h 11min 8s
Wall time: 2h 11min 10s
```

In [101]:

```
clf.get_params
```

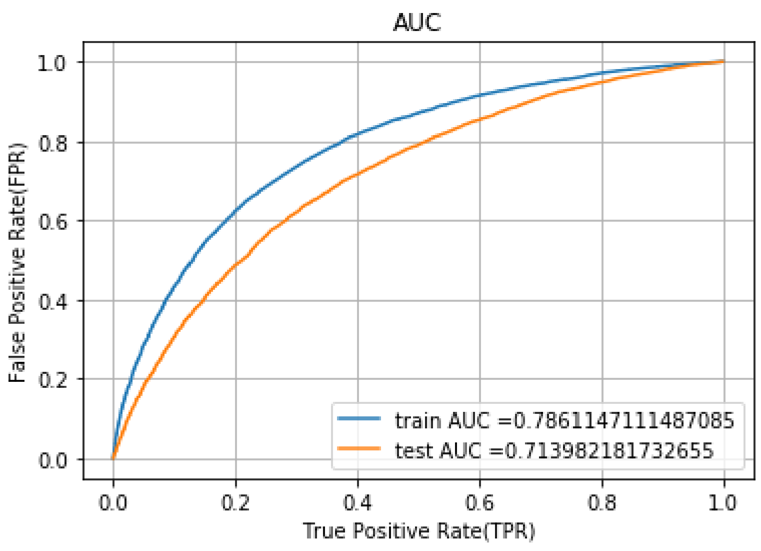
Out[101]:

```
<bound method BaseEstimator.get_params of LogisticRegression(C=1e-06, clas
s_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='warn', n_jobs=None, penalty
='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=
0,
                   warm_start=False)>
```

2.1.2 Train model using the best hyper-parameter(lamda) value set1

In [109]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
LR = LogisticRegression(C= 0.001, class_weight='balanced')
LR.fit(set1_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train, LR.predict_log_proba(set1_train)[
test_fpr, test_tpr , thresholds = roc_curve(y_test, LR.predict_log_proba(set1_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.xscale('log')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



```
CPU times: user 9.28 s, sys: 200 ms, total: 9.48 s
Wall time: 9.11 s
```

2.1.3. Confustion Matrix set1_train and set1_test

In [187]:

```
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.rou
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:predictions.append(0)
    return predictions
```

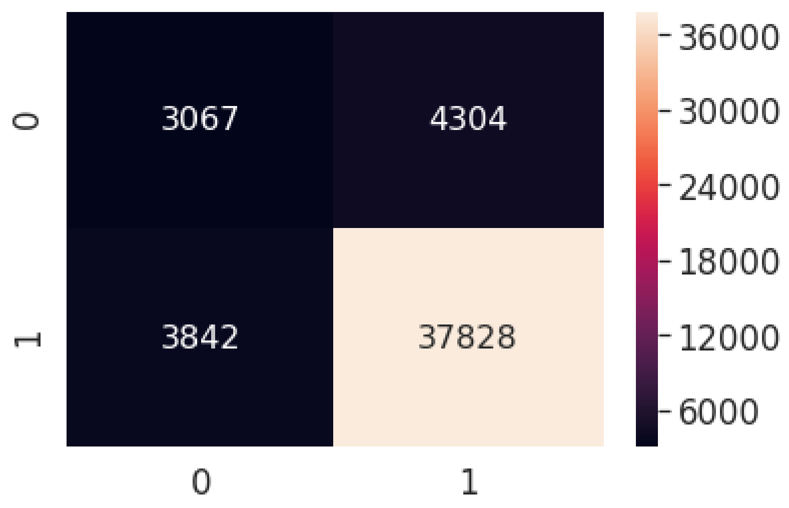
In [112]:

```
#Confustion Matrix Set1_train

LR.fit(set1_train,y_train)
y_train_pred_1 = LR.predict_log_proba(set1_train)[:,1]
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_1,thr


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999539862972 for threshold -0.978
```

Out[112]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f02197302b0>
```



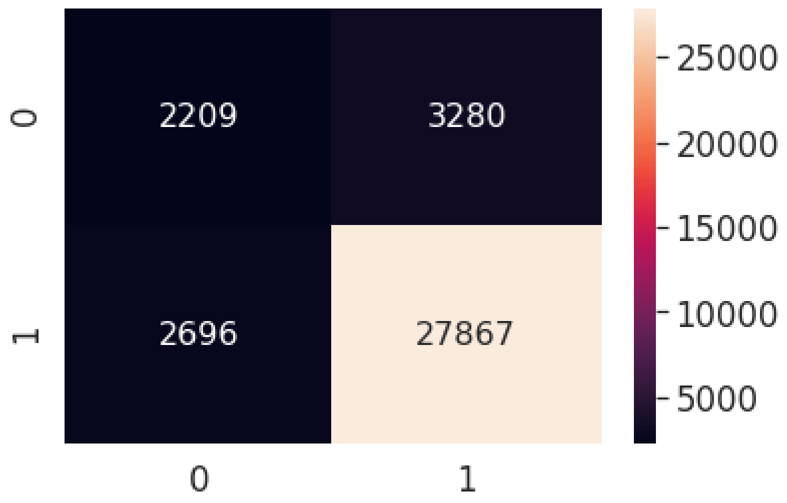
In [113]:

```
#Confustion Matrix Set1_test

LR.fit(set1_test,y_test)
y_test_pred_1 = LR.predict_log_proba(set1_test)[:,1]
conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred_1,thresh


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_test_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999539862972 for threshold -0.978
```

Out[113]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0219774160>
```



2.2 Applying Logistic Regression on TFIDF, SET 2

Merging features encoding numerical + categorical features TFIDF, SET 2

In [116]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set2_ = all categorical features + numarical features + essays_tfidf + project_title_

set2_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     essays_tfidf_train,teacher_number_of_previously_posted_projects_no
                     price_normalizer_train)).tocsr()


set2_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                    teacher_prefix_one_hot_test,project_grade_category_one_hot_test, pr
                    essays_tfidf_test,teacher_number_of_previously_posted_projects_norm
                    price_normalizer_test)).tocsr()

set2_cv  = hstack((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv,school_s
                   teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv,project_
                   essays_tfidf_cv,teacher_number_of_previously_posted_projects_normali
                   price_normalizer_cv)).tocsr()

print("Final Data Matrix of set2 :")
print("shape of set2_train and y_train :", set2_train.shape , y_train.shape)
print("shape of set2_test and y_test   :", set2_test.shape , y_test.shape)
print("shape of set2_cv and y_cv       :", set2_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set2 :
shape of set2_train and y_train : (49041, 9804) (49041,)
shape of set2_test and y_test   : (36052, 9804) (36052,)
shape of set2_cv and y_cv       : (24155, 9804) (24155,)
```

2.2.1.A Hyper parameter Tuning to find alpha:: Simple cross Validation set1

In [118]:

```
%%time
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
params = [0.000001,0.00001,0.001,0.1,1,5,10,50,100]
for i in params:
    # instantiate learning model (C = 10000)
    clf = LogisticRegression(C=i, class_weight='balanced',penalty='l2')
    # fitting the model on crossvalidation train

    clf.fit(set2_train, y_train)
    # predict the response on the crossvalidation train

    pred = clf.predict(set2_cv)
    # evaluate CV accuracy

    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)
    print('\nCV accuracy for lamda = %d is %d%%' % (i, acc))
```

```
CV accuracy for lamda = 0 is 60%

CV accuracy for lamda = 0 is 58%

CV accuracy for lamda = 0 is 57%

CV accuracy for lamda = 0 is 68%

CV accuracy for lamda = 1 is 70%

CV accuracy for lamda = 5 is 70%

CV accuracy for lamda = 10 is 69%

CV accuracy for lamda = 50 is 69%

CV accuracy for lamda = 100 is 69%
CPU times: user 4min 35s, sys: 516 ms, total: 4min 36s
Wall time: 4min 36s
```

In [98]:

```
clf.get_params
```

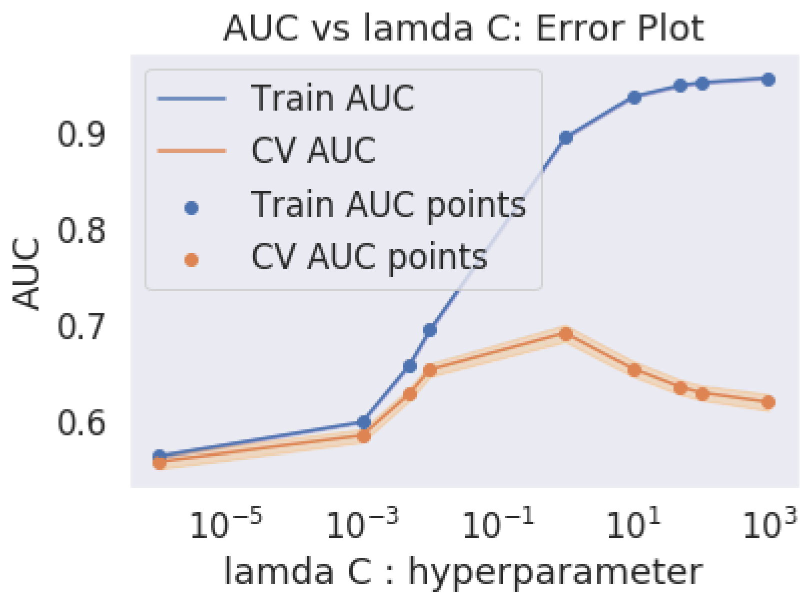
Out[98]:

```
<bound method BaseEstimator.get_params of LogisticRegression(C=1e-06, clas
s_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='warn', n_jobs=None, penalty
='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=
0,
                   warm_start=False)>
```

2.2.1.B Hyper parameter Tuning to find lamda :: RandomizedSearchCV

In [122]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
LR = LogisticRegression(class_weight='balanced')
C = [0.000001, 0.001, 0.005, 0.01, 1, 10, 50,100,1000]
params = {'C':[0.000001, 0.001, 0.005, 0.01, 1, 10, 50,100,1000]}
clf = RandomizedSearchCV(LR, params, cv=5,scoring='roc_auc', return_train_score=True)
clf.fit(set2_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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

plt.plot(params['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['C'],cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(params['C'], train_auc, label='Train AUC points')
plt.scatter(params['C'], cv_auc, label='CV AUC points')

plt.legend()
plt.xscale('log')
plt.xlabel("lamda C : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs lamda C: Error Plot")
plt.grid()
plt.show()
```



```
CPU times: user 26min 5s, sys: 2.7 s, total: 26min 8s
```

```
Wall time: 26min 8s
```

In [123]:

```
clf.get_params
```

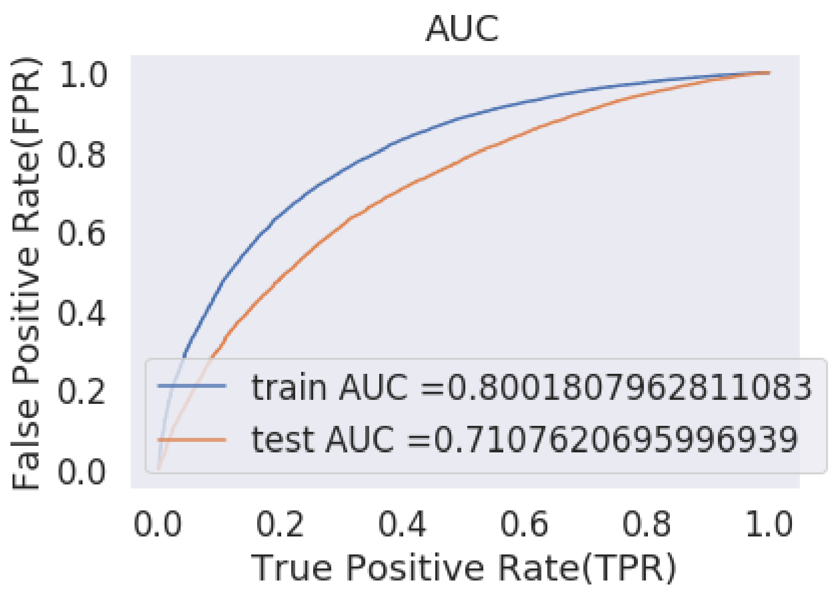
Out[123]:

```
<bound method BaseEstimator.get_params of RandomizedSearchCV(cv=5, error_s
core='raise-deprecating',
                   estimator=LogisticRegression(C=1.0, class_weight='balan
ced',
                                                dual=False, fit_intercept=
True,
                                                intercept_scaling=1,
                                                l1_ratio=None, max_iter=10
0,
                                                multi_class='warn', n_jobs
=None,
                                                penalty='l2', random_state
=None,
                                                solver='warn', tol=0.0001,
                                                verbose=0, warm_start=Fals
e),
                   iid='warn', n_iter=10, n_jobs=None,
                   param_distributions={'C': [1e-06, 0.001, 0.005, 0.01,
1, 10,
                                              50, 100, 1000]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)>
```

2.2.2 Train model using the best hyper-parameter(lamda) value set2

In [206]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
LR = LogisticRegression(C= 0.1,class_weight='balanced')
LR.fit(set2_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train, LR.predict_log_proba(set2_train)[
test_fpr, test_tpr , thresholds = roc_curve(y_test, LR.predict_log_proba(set2_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.xscale('log')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



```
CPU times: user 8.49 s, sys: 168 ms, total: 8.66 s
Wall time: 8.29 s
```

2.2.3 Confusion Matrix Set2_train and test

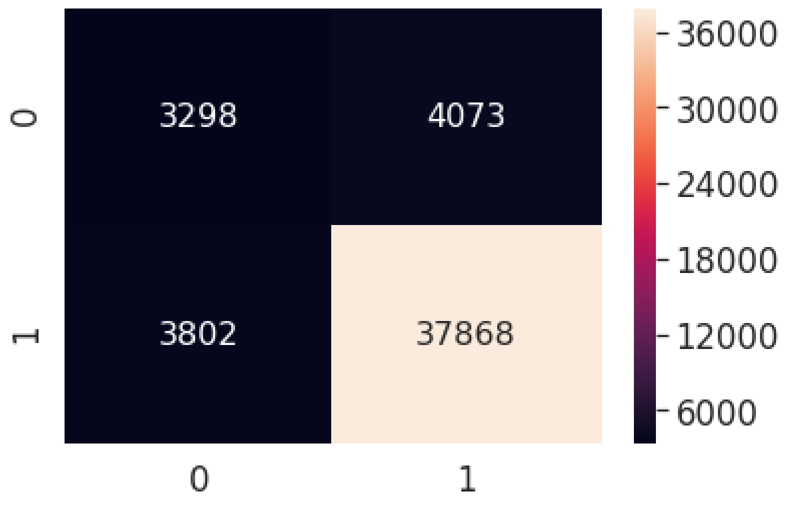
In [207]:

```
# Confusion Matrix Set2_train
LR.fit(set2_train,y_train)
y_train_pred_2 = LR.predict_log_proba(set2_train)[:,1]
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_2,thr
range(2))

sns.set(font_scale=1.5)
sns.heatmap(conf_matr_df_train_2,annot=True,annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999539862972 for threshold -0.931
```

Out[207]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01d825eb70>
```



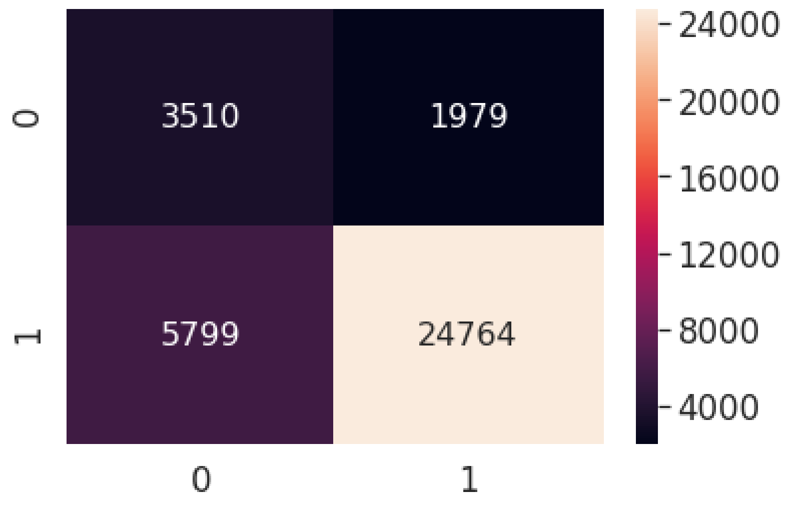
In [208]:

```
# Confusion Matrix Set2_test
LR.fit(set2_test,y_test)
y_test_pred_2 = LR.predict_log_proba(set2_test)[:,1]
conf_matr_df_test_2 =pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred_2,thresho
range(2))

sns.set(font_scale=1.5)
sns.heatmap(conf_matr_df_test_2,annot=True,annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2499999917023799 for threshold -0.772
```

Out[208]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01d81c7470>
```



2.3 Apply Logestic Regression on set3 AVG W2V

Merging features encoding numerical + categorical features AVG W2V, SET3

In [131]:

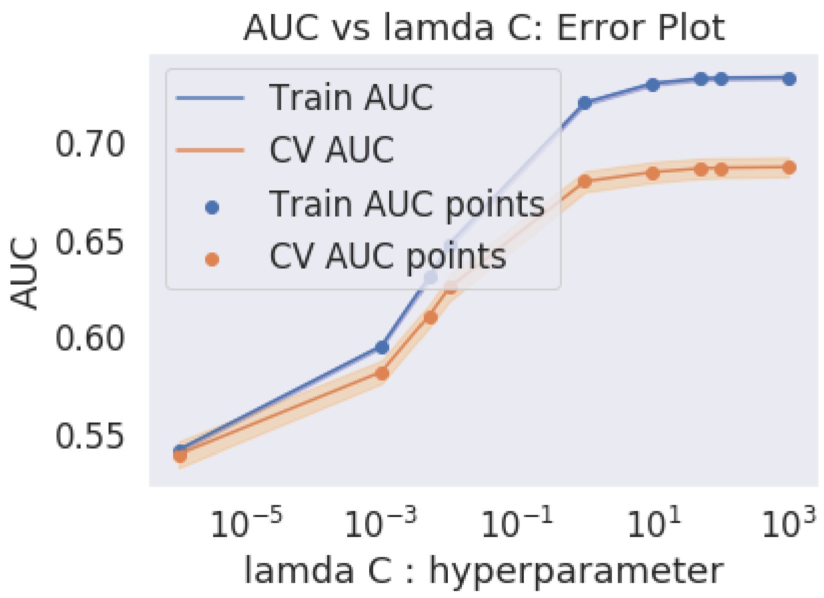
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set3_ = all categorical features + numarical features + essays_avg_w2v_vectors , proj


set3_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     project_title_avg_w2v_vectors_train,teacher_number_of_previously_p
                     price_normalizer_train))

set3_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                     teacher_prefix_one_hot_test,project_grade_category_one_hot_test, e
                     project_title_avg_w2v_vectors_test,teacher_number_of_previously_po
                     price_normalizer_test))

set3_cv = hstack((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv, school_s
                     teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv, essay
                     project_title_avg_w2v_vectors_cv,teacher_number_of_previously_post
                     price_normalizer_cv))

print("Final Data Matrix of set3 :")
print("shape of set3_train and y_train :", set3_train.shape , y_train.shape)
print("shape of set3_test and y_test   :", set3_test.shape , y_test.shape)
print("shape of set3_cv and y_cv       :", set3_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set3 :
shape of set3_train and y_train : (49041, 701) (49041,)
shape of set3_test and y_test   : (36052, 701) (36052,)
shape of set3_cv and y_cv       : (24155, 701) (24155,)
```

2.3.1 Hyperparameter Tuning to find best lamda :: RandomizedsearchCV

In [132]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
LR = LogisticRegression(class_weight='balanced')
C = [0.000001, 0.001, 0.005, 0.01, 1, 10, 50,100,1000]
params = {'C':[0.000001, 0.001, 0.005, 0.01, 1, 10, 50,100,1000]}
clf = RandomizedSearchCV(LR, params, cv=5,scoring='roc_auc', return_train_score=True)
clf.fit(set3_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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

plt.plot(params['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['C'],cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(params['C'], train_auc, label='Train AUC points')
plt.scatter(params['C'], cv_auc, label='CV AUC points')

plt.legend()
plt.xscale('log')
plt.xlabel("lamda C : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs lamda C: Error Plot")
plt.grid()
plt.show()
```

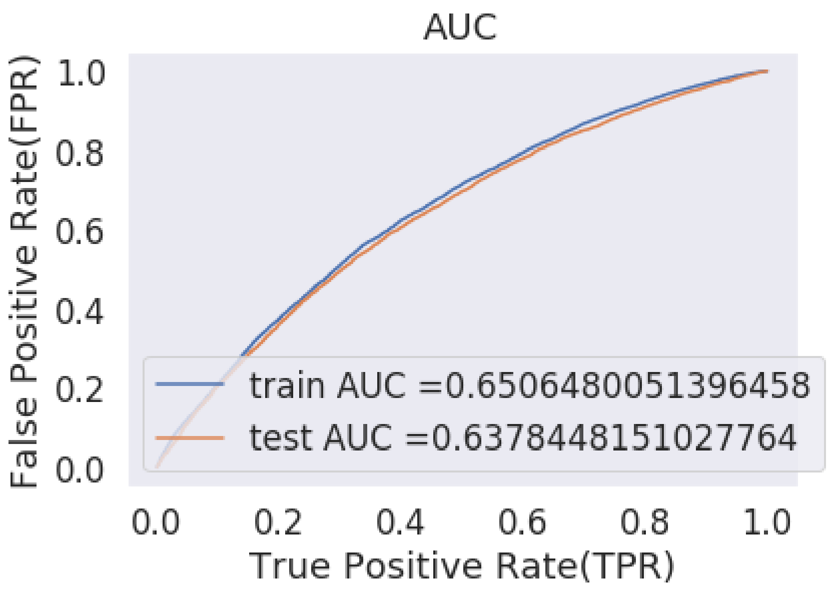


```
CPU times: user 53min 39s, sys: 13.5 s, total: 53min 53s
Wall time: 53min 53s
```

2.3.2 Train model using the best hyper-parameter value set3

In [133]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
LR = LogisticRegression(C= 0.01, class_weight='balanced')
LR.fit(set3_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train, LR.predict_log_proba(set3_train)[
test_fpr, test_tpr , thresholds = roc_curve(y_test, LR.predict_log_proba(set3_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.xscale('log')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



```
CPU times: user 14.7 s, sys: 328 ms, total: 15.1 s
Wall time: 14.7 s
```

2.3.3. Confusion Matrix of set3 _train and test

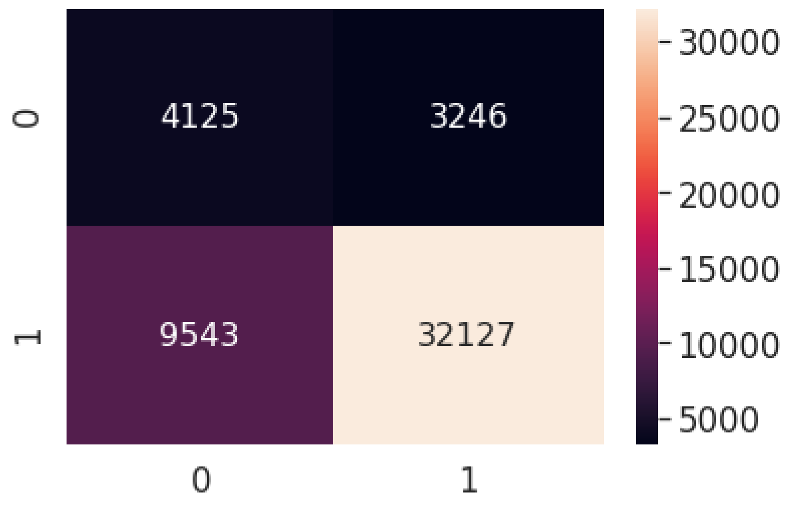
In [219]:

```
# Confusion Matrix set3_train
LR.fit(set3_train,y_train)
y_train_pred_3 = LR.predict_log_proba(set3_train)[:,1]
confusion_matr_df_train_3 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred_

sns.set(font_scale=1.5) # for label Size
sns.heatmap(confusion_matr_df_train_3, annot=True, annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999539862972 for threshold -0.827
```

Out[219]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01d7f8e898>
```



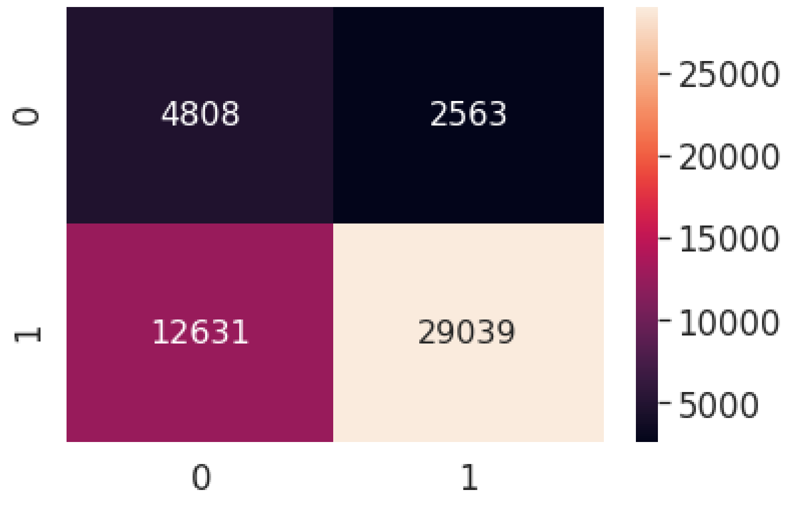
In [220]:

```
# Confusion Matrix set3_test
LR.fit(set3_test, y_test)
y_test_pred_3 = LR.predict_log_proba(set3_test)[:,1]
confusion_matr_df_test_3 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_3

sns.set(font_scale=1.5) # for label size
sns.heatmap(confusion_matr_df_test_3, annot=True, annot_kws={"size" :16}, fmt = 'g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999170237988 for threshold -0.733
```

Out[220]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01d7f5ae80>
```



2.4 Applying Logistic Regresion on TFIDF W2V, SET 4

Merging features encoding numerical + categorical features TFIDF w2V set4

In [147]:

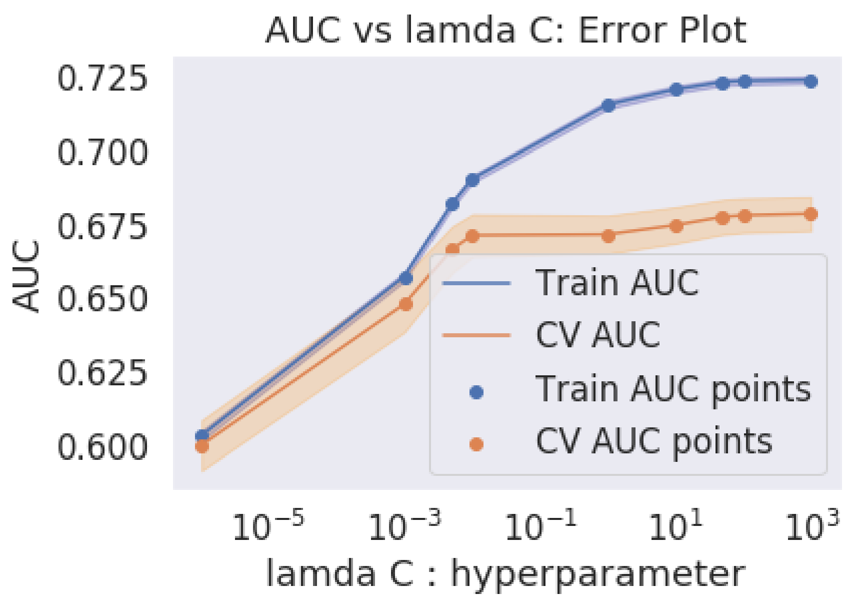
```
# Merging two sparse matrixs : https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matrix
# set_4 = encoded numarical + categorical + project_title_tfidf_w2v_vectors +essays_tfi

set4_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     preprocessed_essays_train_tfidf_w2v_vectors, preprocessed_project_
                     teacher_number_of_previously_posted_projects_normalizer_train,pric

set4_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                   teacher_prefix_one_hot_test, project_grade_category_one_hot_test,
                   preprocessed_project_title_test_tfidf_w2v_vectors,preprocessed_essay
                   teacher_number_of_previously_posted_projects_normalizer_test, price_

set4_cv = hstack((clean_categories_one_hot_cv,clean_subcategories_one_hot_cv, school_st
                 teacher_prefix_one_hot_cv, project_grade_category_one_hot_cv,
                 preprocessed_project_title_cv_tfidf_w2v_vectors,preprocessed_essays_cv
                 teacher_number_of_previously_posted_projects_normalizer_cv, price_norm

print("Final Data Matrix of set4 :")
print("shape of set4_train and y_train :", set4_train.shape , y_train.shape)
print("shape of set4_test and y_test   :", set4_test.shape , y_test.shape)
print("shape of set4_cv and y_cv       :", set4_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set4 :
shape of set4_train and y_train : (49041, 701) (49041,)
shape of set4_test and y_test   : (36052, 701) (36052,)
shape of set4_cv and y_cv       : (24155, 701) (24155,)
```

2.4.1 Hyper parameter Tuning to find lamda : RandomizedSearchCV

In [148]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
LR = LogisticRegression(class_weight='balanced')
C = [0.000001, 0.001, 0.005, 0.01, 1, 10, 50,100,1000]
params = {'C':[0.000001, 0.001, 0.005, 0.01, 1, 10, 50,100,1000]}
clf = RandomizedSearchCV(LR, params, cv=5,scoring='roc_auc', return_train_score=True)
clf.fit(set4_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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

plt.plot(params['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['C'],cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(params['C'], train_auc, label='Train AUC points')
plt.scatter(params['C'], cv_auc, label='CV AUC points')

plt.legend()
plt.xscale('log')
plt.xlabel("lamda C : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs lamda C: Error Plot")
plt.grid()
plt.show()
```



```
CPU times: user 24min 52s, sys: 3.54 s, total: 24min 55s
Wall time: 24min 55s
```

```
In [174]:
```

```
clf.get_params
```

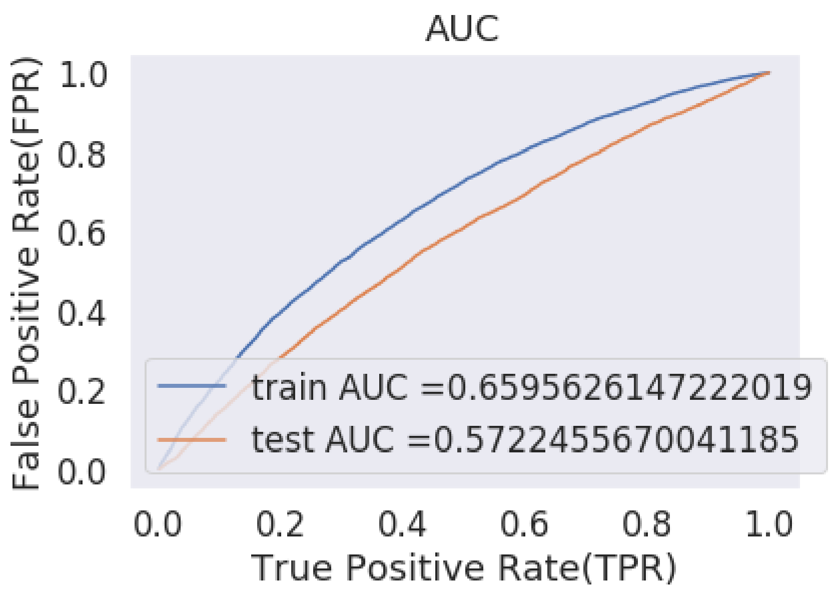
```
Out[174]:
```

```
<bound method BaseEstimator.get_params of RandomizedSearchCV(cv=5, error_s
core='raise-deprecating',
                   estimator=LogisticRegression(C=1.0, class_weight='balan
ced',
                                                dual=False, fit_intercept=
True,
                                                intercept_scaling=1,
                                                l1_ratio=None, max_iter=10
0,
                                                multi_class='warn', n_jobs
=None,
                                                penalty='l2', random_state
=None,
                                                solver='warn', tol=0.0001,
                                                verbose=0, warm_start=Fals
e),
                   iid='warn', n_iter=10, n_jobs=None,
                   param_distributions={'C': [1e-06, 0.001, 0.005, 0.01,
1, 10,
                                              50, 100, 1000]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)>
```

2.4.2 Train model using best hyperparameter value set4

```
In [173]:
```

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
LR = LogisticRegression(C= 0.001, class_weight='balanced')
LR.fit(set4_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train, LR.predict_log_proba(set4_train)[
test_fpr, test_tpr , thresholds = roc_curve(y_test, LR.predict_log_proba(set4_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.xscale('log')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



```
CPU times: user 3.32 s, sys: 224 ms, total: 3.54 s
Wall time: 3.14 s
```

2.4.3 Confusion Matrix set4_train and test

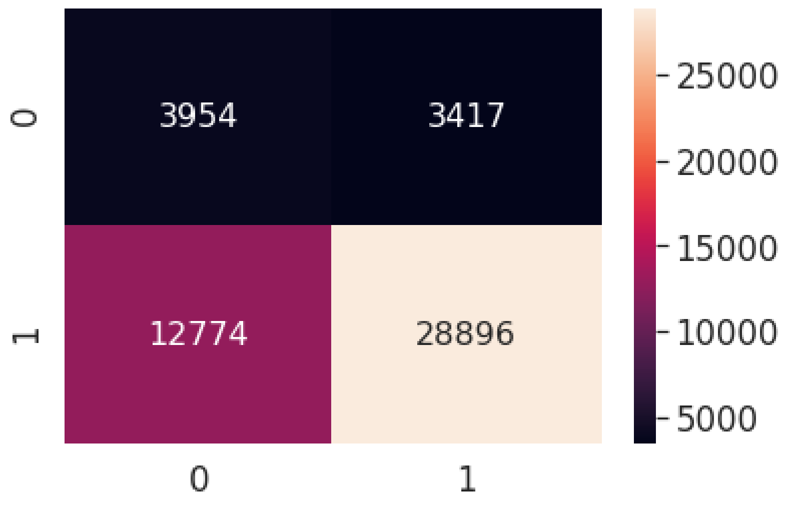
In [179]:

```
# confusion Matrix set4_train
LR.fit(set4_train, y_train)
y_train_pred_4 = LR.predict_log_proba(set4_train)[:,1]
confusion_matr_df_train_4 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_

sns.set(font_scale=1.5) # font size
sns.heatmap(confusion_matr_df_train_4, annot=True, annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999539862972 for threshold -0.732
```

Out[179]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01d8677828>
```



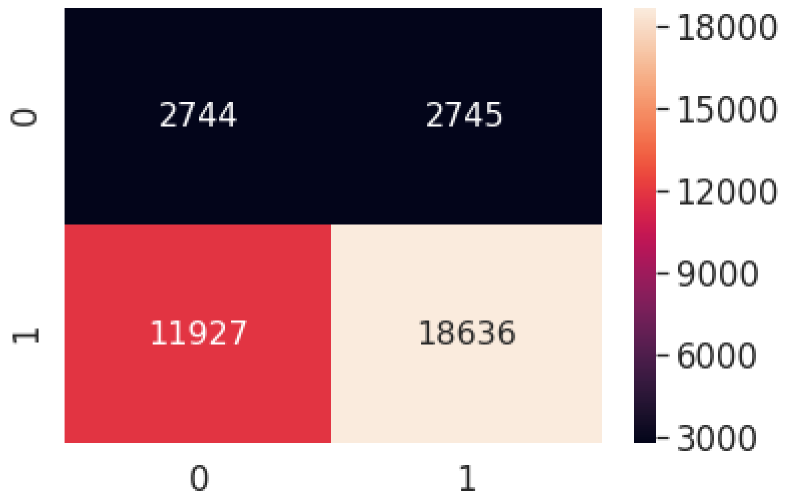
In [181]:

```
# Confusion Matrix set4_test
y_test_pred_4 = LR.predict_log_proba(set4_test)[:,1]
confusion_matr_df_test_4 = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred_4, 

sns.set(font_scale=1.5) # for font size
sns.heatmap(confusion_matr_df_test_4, annot=True, annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2499999917023799 for threshold -0.698
```

Out[181]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f01d8638470>
```



2.5 Logistic Regression with added Features `Set 5`

Merging features encoding numerical + categorical features set5

In [174]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstcj=k function we are concatinating a sparse matrix and dense matrix
# set_5 = selected numarical + selected catogorical features

set5_train = hstack ((clean_categories_one_hot_train, clean_subcategories_one_hot_train
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     teacher_number_of_previously_posted_projects_normalizer_train,pric

set5_test  = hstack ((clean_categories_one_hot_test, clean_subcategories_one_hot_test, 
                     teacher_prefix_one_hot_test, project_grade_category_one_hot_test,
                     teacher_number_of_previously_posted_projects_normalizer_test, pric

set5_cv    = hstack ((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv, schoo
                     teacher_prefix_one_hot_cv, project_grade_category_one_hot_cv,
                      teacher_number_of_previously_posted_projects_normalizer_cv,price_


print("Final data matrix of set_5")
print("Shape of set5_train and y_train :", set5_train.shape, y_train.shape)
print("Shape of set5_test and y_test   :", set5_test.shape, y_train.shape)
print("Shape of set5_cv and y_cv       :", set5_cv.shape, y_cv.shape)
```

```
Final data matrix of set_5
Shape of set5_train and y_train : (49041, 102) (49041,)
Shape of set5_test and y_test   : (36052, 102) (49041,)
Shape of set5_cv and y_cv       : (24155, 102) (24155,)
```

2.5.1.A Hyperparameter tuning to find best lamda : Simple cross validation set5

In [175]:

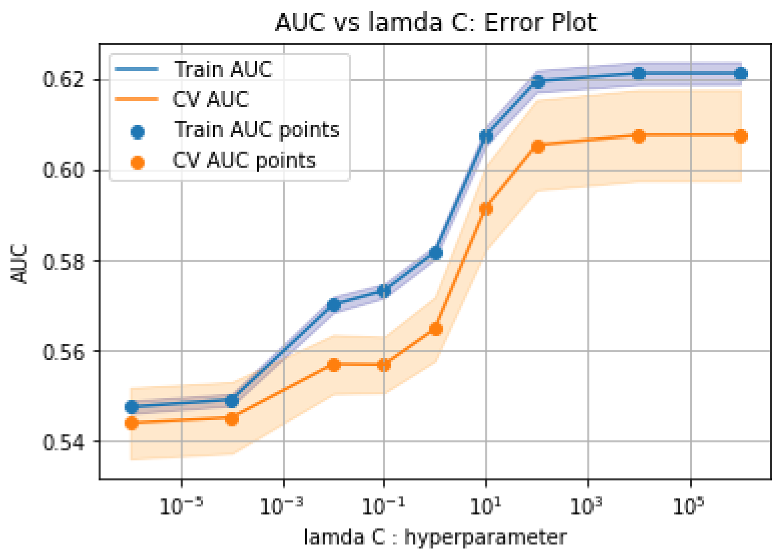
```
%%time
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
params = [0.000001,0.00001,0.001,0.1,10,100,1000, 10000]
for i in params:
    # instantiate learning model
    clf = LogisticRegression(C=i, class_weight='balanced',penalty='l2')
    # fitting the model on crossvalidation train

    clf.fit(set5_train, y_train)
    # predict the response on the crossvalidation train

    pred = clf.predict(set5_cv)
    # evaluate CV accuracy

    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)
    print('\nCV accuracy for lamda = %d is %d%%' % (i, acc))
```

```

CV accuracy for lamda = 0 is 59%

CV accuracy for lamda = 0 is 58%

CV accuracy for lamda = 0 is 56%

CV accuracy for lamda = 0 is 54%

CV accuracy for lamda = 10 is 54%

CV accuracy for lamda = 100 is 53%

CV accuracy for lamda = 1000 is 53%

CV accuracy for lamda = 10000 is 53%
Wall time: 16.6 s
```

2.5.1.B Hyperparameter tuning to find best lamda : RandomizedsearchCV set5

In [176]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
LR = LogisticRegression(class_weight='balanced')
C = [10**-6, 10**-4, 10**-2, 10**-1, 1 , 10, 10**2,  10**4,  10**6]
params = {'C':[10**-6, 10**-4, 10**-2, 10**-1, 1 , 10, 10**2,  10**4,  10**6]}
clf = RandomizedSearchCV(LR, params, cv=5,scoring='roc_auc', return_train_score=True)
clf.fit(set5_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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

plt.plot(params['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['C'],cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(params['C'], train_auc, label='Train AUC points')
plt.scatter(params['C'], cv_auc, label='CV AUC points')

plt.legend()
plt.xscale('log')
plt.xlabel("lamda C : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs lamda C: Error Plot")
plt.grid()
plt.show()
```



```
Wall time: 1min 15s
```

In [177]:

```
LR.score
```

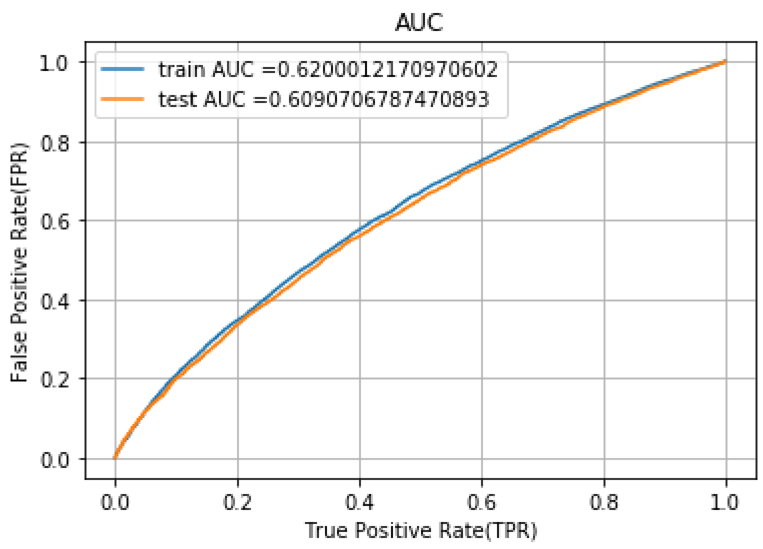
Out[177]:

```
<bound method ClassifierMixin.score of LogisticRegression(C=1.0, class_wei
ght='balanced', dual=False,
          fit_intercept=True, intercept_scaling=1, max_iter=100,
          multi_class='warn', n_jobs=None, penalty='l2', random_state=Non
e,
          solver='warn', tol=0.0001, verbose=0, warm_start=False)>
```

2.5.2 traing model using best hyperparameter set_5

In [183]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
LR = LogisticRegression(C= 1000, class_weight='balanced')
LR.fit(set5_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train, LR.predict_log_proba(set5_train)[
test_fpr, test_tpr , thresholds = roc_curve(y_test, LR.predict_log_proba(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.xscale('log')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



```
Wall time: 4.09 s
```

2.5.3 Confusion Matrix set5_train and test

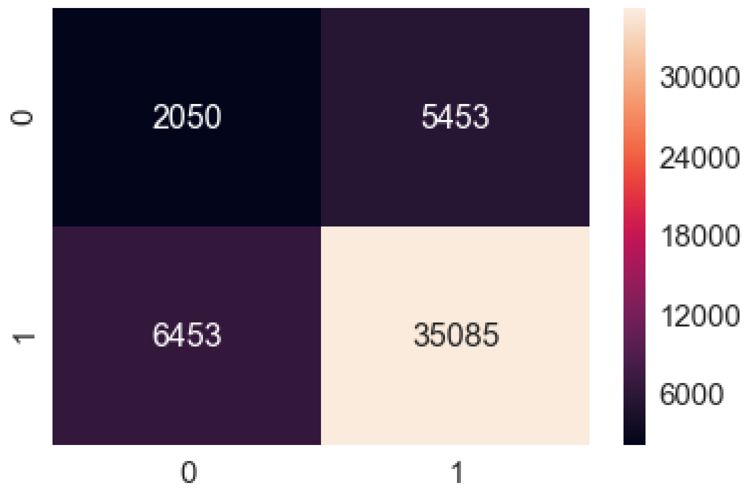
In [188]:

```
# Confusion matrix set5_train
y_train_pred_5 = LR.predict_log_proba(set5_train)[:,1]
confusion_matr_df_train_5 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_

sns.set(font_scale= 1.5) # for font size
sns.heatmap(confusion_matr_df_train_5, annot=True, annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999555910898 for threshold -0.831
```

Out[188]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x5a225208>
```



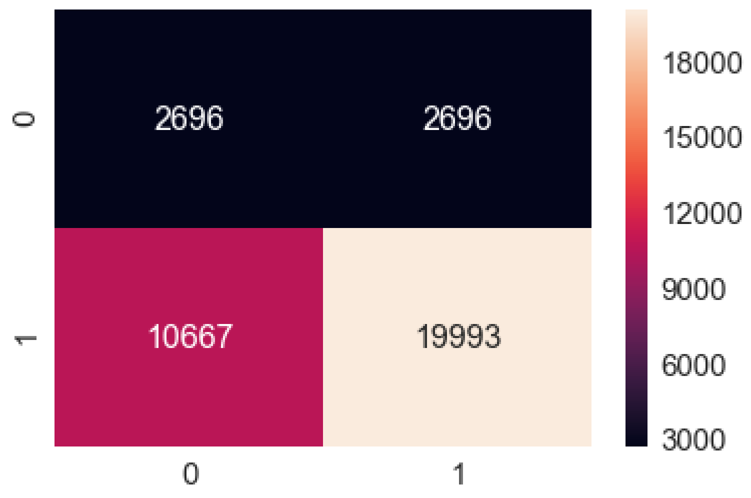
In [189]:

```
# Confusion Matrix set5_test
y_test_pred_5 = LR.predict_log_proba(set5_test)[:,1]
confusion_matr_df_test_5 = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred_5,

sns.set(font_scale=1.5) # for font size
sns.heatmap(confusion_matr_df_test_5, annot = True, annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold -0.732
```

Out[189]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x51a0dc18>
```



3. Conclusion

In [3]:

```python
# Link : http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
p = PrettyTable()

p.field_names = ["Vectorizer", "Model", "Hyper parameter(lamda/C)", "AUC"]
p.add_row(["BOW","Logistic Regression",0.001, 0.71])
p.add_row(["TFIDF","Logistic Regression", 0.1, 0.71])
p.add_row(["Avg W2V", "Logistic Regression", 0.01, 0.63])
p.add_row(["TFIDF W2V", "Logistic Regression", 0.001, 0.57])
p.add_row(["added Features Set 5", "Logistic Regression", 1000,0.60])

print(p)
```

```
+----------------------+---------------------+--------------------------+-
-----+
|      Vectorizer      |        Model        | Hyper parameter(lamda/C) |
AUC  |
+----------------------+---------------------+--------------------------+-
-----+
|         BOW          | Logistic Regression |          0.001           |
0.71 |
|        TFIDF         | Logistic Regression |           0.1            |
0.71 |
|       Avg W2V        | Logistic Regression |           0.01           |
0.63 |
|      TFIDF W2V       | Logistic Regression |          0.001           |
0.57 |
| added Features Set 5 | Logistic Regression |           1000           |
0.6  |
+----------------------+---------------------+--------------------------+-
-----+
```

Thank You.

Sign Off Ramesh Battu (https://www.linkedin.com/in/rameshbattuai/)