Document Created Date:25-02-2019

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. **Example**
`project_title`	Title of the project • `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 **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 **E**

* 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 Naive Bayes
- 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(GridSearchCV- 2nd technique)
- Train the Naive Bayes model using best hyperparameter and ploting auc roc-curve
- Ploting confusion matrix(heatmaps)
- Finding top 10 important features of positive class and negative class
- Observation on overall model performences
- Ploting the performences by tableu format.

C:\Users\Ramesh Battu> import required libraries

In [176]:

```
%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 [2]:

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

In [3]:

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

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

In [4]:

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


#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[4]:

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

In [5]:

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

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

Out[5]:

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

1.1.1 preprocessing of project_subject_categories

In [6]:

```
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 return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.1.2 preprocessing of project_subject_subcategories

In [7]:

```
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 [8]:

```
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-pyth
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 "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['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 [9]:

```
# citation code :https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['teacher_prefix'].value_counts().index[0
teacher_prefix_catogories = list(project_data['teacher_prefix'].values)
# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit
# 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 sp
        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 [10]:

```
# 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 "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['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(), k
```

1.2. Text Preprocessing

1.2.1 Text Preprocessing of essay

In [11]:

```
# 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 [12]:

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

Out[12]:

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

In [13]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*125)
print(project_data['essay'].values[250])
print("="*125)
print(project_data['essay'].values[1000])
print("="*125)
print(project_data['essay'].values[30000])
print("="*125)
print(project_data['essay'].values[99999])
print("="*125)
```

```
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 [14]:

```
# 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 [15]:

```
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 [16]:

```
# \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 [17]:

```
#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 [18]:

```
# 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","elsewh
            "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 [19]:

```
%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 [05:56<00:00, 306.11
it/s]
```

In [20]:

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

Out[20]:

```
'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 [21]:

```
print(project_data["project_title"].head(1))
```

```
55660    Engineering STEAM into the Primary Classroom
Name: project_title, dtype: object
```

In [22]:

```
# Printing some random variables
print(project_data["project_title"].values[0])
print("="*125)
print(project_data["project_title"].values[400])
print("="*125)
print(project_data["project_title"].values[2000])
print("="*125)
print(project_data["project_title"].values[30000])
print("="*125)
print(project_data["project_title"].values[99999])
print("="*125)
```

```
Engineering STEAM into the Primary Classroom
=============================================================================
==================================================
Desks and Chairs for Autism Classroom!
=============================================================================
==================================================
Empowering Students through Art in the Makerspace
=============================================================================
==================================================
Wiggly Without Consequences
=============================================================================
==================================================
Turning to Flexible Seating: One Sixth-Grade Class's Journey to Freedom
=============================================================================
==================================================
```

In [23]:

```
# 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 [24]:

```
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 [25]:

```
# \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 [26]:

```
#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 [27]:

```
# 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","elsewh
            "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 [28]:

```
%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, 8786.44
it/s]
```

In [29]:

```
preprocessed_project_title[99999]
```

Out[29]:

```
'turning flexible seating sixth grade class journey freedom'
```

1.3. Numerical normalization

1.3.1 normalization_price

In [30]:

```
# 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[30]:

```
(109248, 20)
```

In [31]:

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

Out[31]:

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

In [32]:

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

```
(109248,)
```

In [33]:

```
# 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 [34]:

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

Out[34]:

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

In [35]:

```
# 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_numb
```

```
[[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 [36]:

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

Out[36]:

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

In [37]:

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

Out[37]:

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

In [38]:

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

In [39]:

```
# 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 [40]:

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

clean_categories_one_hot_train = vectorizer.fit_transform(X_train['clean_categories'].va
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.sh
print("Shape of matrix after one hot encodig test  : ",clean_categories_one_hot_test.sha
print("Shape of matrix after one hot encodig cv    : ",clean_categories_one_hot_cv.shape
```

```
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 [41]:

```
# we count vectorizer to conver the values into one hot encoded features
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_subcategor
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.sh
```

```
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 [42]:

```
# we count vectorizer to conver the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer (vocabulary=list(sorted_school_state_dict.keys()), lowerca

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("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 [43]:

```
# 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.as
teacher_prefix_one_hot_cv    = vectorizer.transform (X_cv['teacher_prefix'].values.asty

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)
```

1.4.5 Vectorization of project_grade_category

In [45]:

```
# we use count vectorizer to convert 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_cate
project_grade_category_one_hot_cv    = vectorizer.transform (X_cv['project_grade_catego

print(vectorizer.get_feature_names())

print("-------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",project_grade_category_one_hot_t
print("Shape of matrix after one hot encodig test  : ",project_grade_category_one_hot_t
print("Shape of matrix after one hot encodig cv    : ",project_grade_category_one_hot_c
```

```
['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_ train, test cv

In [46]:

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

essays_bow_train = vectorizer.fit_transform(X_train['essay'].values)
essays_bow_test = vectorizer.transform(X_test['essay'].values)
essays_bow_cv = vectorizer.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, 12579)
Shape of matrix after one hot encodig test  :  (36052, 12579)
Shape of matrix after one hot encodig cv    :  (24155, 12579)
```

1.5.1.1 Vectorization of essays tfidf _ train, test cv

In [47]:

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

essays_tfidf_train = vectorizer.fit_transform(X_train['essay'].values)
essays_tfidf_test = vectorizer.transform(X_test['essay'].values)
essays_tfidf_cv = vectorizer.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, 12579)
Shape of matrix after one hot encodig test     :  (36052, 12579)
Shape of matrix after one hot encodig cv       :  (24155, 12579)
```

1.5.2 Vectorization of project_title bow train, test, cv

In [48]:

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

project_title_bow_train = vectorizer.fit_transform(X_train['project_title'].values)
project_title_bow_test = vectorizer.transform(X_test['project_title'].values)
project_title_bow_cv = vectorizer.transform(X_cv['project_title'].values)

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, 2116)
Shape of matrix after one hot encodig test  :  (36052, 2116)
Shape of matrix after one hot encodig cv    :  (24155, 2116)
```

1.5.2.1 Vectorization of project_title tfidf train, test. cv

In [49]:

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

project_title_tfidf_train = vectorizer.fit_transform(X_train['project_title'].values)
project_title_tfidf_test = vectorizer.transform(X_test['project_title'].values)
project_title_tfidf_cv = vectorizer.transform(X_cv['project_title'].values)

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, 2116)
Shape of matrix after one hot encodig test     :  (36052, 2116)
Shape of matrix after one hot encodig cv       :  (24155, 2116)
```

1.6. Vectorizing Numerical features

1.6.1 Normalization of price_train_test_cv

In [50]:

```
# 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)
----------------------------
[[0.0022698 ]
 [0.00088171]
 [0.00444434]
 ...
 [0.00263755]
 [0.00353701]
 [0.00258881]]
shape of price_normalizer_test : (36052, 1)
----------------------------
[[0.0057053 ]
 [0.0006654 ]
 [0.00494895]
 ...
 [0.00217555]
 [0.00627287]
 [0.00142075]]
shape of price_normalizer_cv   : (24155, 1)
----------------------------
[[0.00342725]
 [0.00132469]
 [0.0002701 ]
 ...
 [0.00261316]
 [0.00395393]
 [0.00281283]]
```

1.6.2 Teacher_number_of_previously_posted_projects_train_test_cv : Numerical / Normalization

In [51]:

```
# 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
_transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1)

teacher_number_of_previously_posted_projects_normalizer_test  = teacher_number_of_previ
.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

teacher_number_of_previously_posted_projects_normalizer_cv    = teacher_number_of_previ
.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))


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_
_normalizer_train.shape)
print("----------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_train)

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

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

```
shape of teacher_number_of_previously_posted_projects_normalizer_train:
(49041, 1)
----------------------------
[[0.00136098]
 [0.00015122]
 [0.00030244]
 ...
 [0.00272196]
 [0.00136098]
 [0.00257074]]
shape of teacher_number_of_previously_posted_projects_normalizer_test :
(36052, 1)
----------------------------
[[0.        ]
 [0.        ]
 [0.00035875]
 ...
 [0.00143499]
 [0.        ]
 [0.04143538]]
shape of teacher_number_of_previously_posted_projects_normalizer_cv   :
```

```
(24155, 1)
---------------------------
[[0.        ]
 [0.00145735]
 [0.        ]
 ...
 [0.02123562]
 [0.00020819]
 [0.        ]]
```

In [52]:

```
project_data.columns
```

Out[52]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'clean_subcategories', 'essay', 'price', 'quantity'],
      dtype='object')
```

we are going to consider

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

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

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

Assignment : Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

2. **The hyper paramter tuning(find best Alpha)**
    - 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
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001

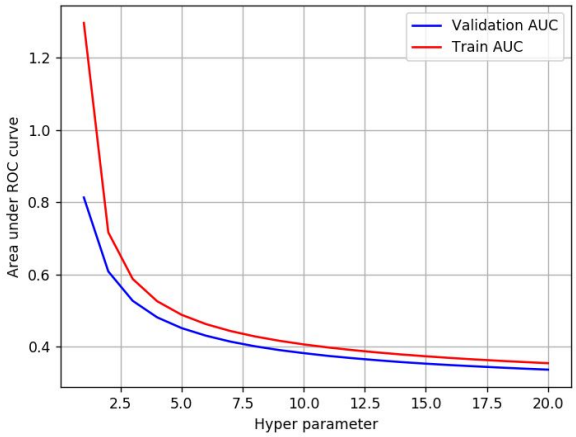
- Find the best hyper paramter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Feature importance**

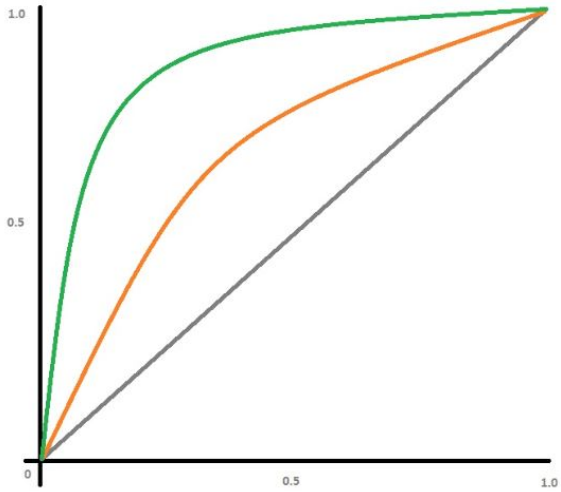
- Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names

4. **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. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.



- 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/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

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

(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

5. **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/)

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

2. Naive Bayes

2.1 Applying Naive Bayes on BOW, SET 1

Merging features encoding numerical + categorical features BOW, SET 1

In [53]:

```
# 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 + essays_bow + project_title_bo

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, 14796) (49041,)
shape of set1_test and y_test   : (36052, 14796) (36052,)
shape of set1_cv and y_cv       : (24155, 14796) (24155,)
```

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

In [54]:

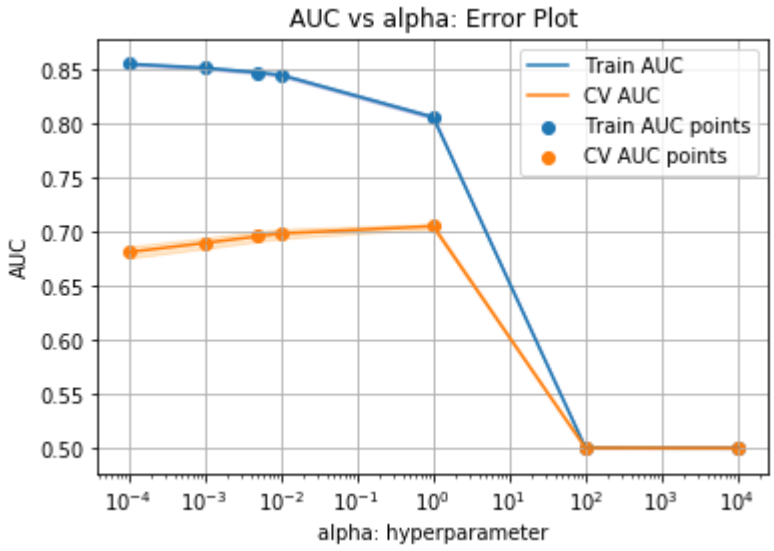
```
%%time
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
for i in range(1,10000,1000):
    # instantiate learning model (alpha = 10000)
    nb = MultinomialNB(alpha=i, class_prior=[0.5,0.5])
    # fitting the model on crossvalidation train

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

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

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

```
CV accuracy for alpha = 1 is 69%

CV accuracy for alpha = 1001 is 84%

CV accuracy for alpha = 2001 is 84%

CV accuracy for alpha = 3001 is 84%

CV accuracy for alpha = 4001 is 84%

CV accuracy for alpha = 5001 is 84%

CV accuracy for alpha = 6001 is 84%

CV accuracy for alpha = 7001 is 84%

CV accuracy for alpha = 8001 is 84%

CV accuracy for alpha = 9001 is 84%
Wall time: 2.09 s
```

2.1.1.B Hyper parameter Tuning to find alpha:: GridSearch cv

In [55]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
nb = MultinomialNB(class_prior=[0.5,0.5])
p = [0.0001, 0.001, 0.005, 0.01, 1, 100, 10000]
params = {'alpha':[0.0001, 0.001, 0.005, 0.01, 1, 100, 10000]}
clf = GridSearchCV(nb, 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['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['alpha'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

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

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

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



Wall time: 15.5 s

In [56]:

```
clf.get_params
```

Out[56]:

```
<bound method BaseEstimator.get_params of GridSearchCV(cv=5, error_score
='raise-deprecating',
       estimator=MultinomialNB(alpha=1.0, class_prior=[0.5, 0.5], fit_prio
r=True),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'alpha': [0.0001, 0.001, 0.005, 0.01, 1, 100, 10000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='roc_auc', verbose=0)>
```

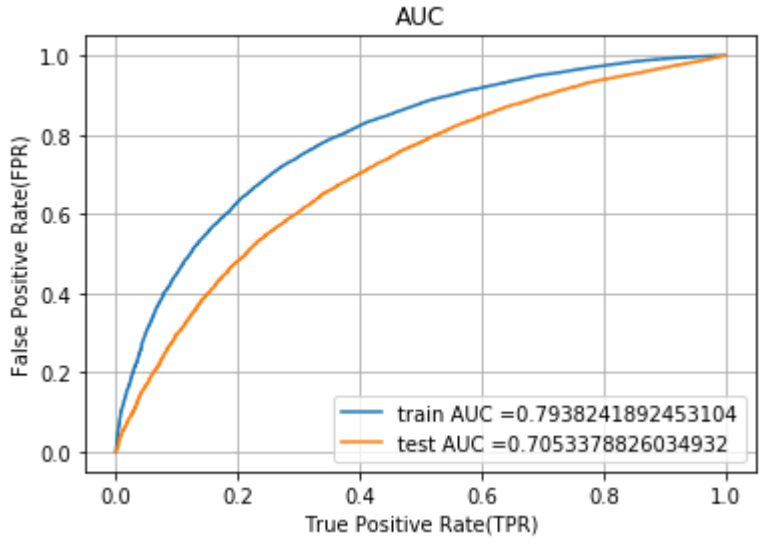
In [57]:

```
best_alpha = 1
```

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

In [58]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_curve, auc
nb = MultinomialNB(alpha=1, class_prior=[0.5,0.5])
nb.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, nb.predict_log_proba(set1_train)[
test_fpr, test_tpr, thresholds = roc_curve(y_test, nb.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()
```




```
Wall time: 629 ms
```

In [59]:

```
print(clf.get_params)
```

```
<bound method BaseEstimator.get_params of GridSearchCV(cv=5, error_score
='raise-deprecating',
       estimator=MultinomialNB(alpha=1.0, class_prior=[0.5, 0.5], fit_prio
r=True),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'alpha': [0.0001, 0.001, 0.005, 0.01, 1, 100, 10000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='roc_auc', verbose=0)>
```

2.1.3 Confustion Matrix set1_train and set1_test

In [60]:

```
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
```

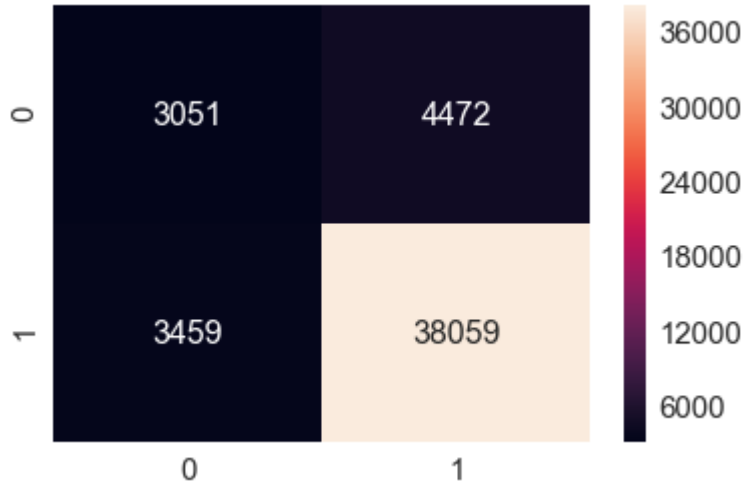
In [61]:

```
# Confustion Matrix Set1_train
y_train_pred = nb.predict_log_proba(set1_train)[:,1]
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred,thres
train_fpr, train_fpr)), range(2),range(2))

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) 0.2499999602442094 for threshold -4.901
```

Out[61]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x274ddb70>
```



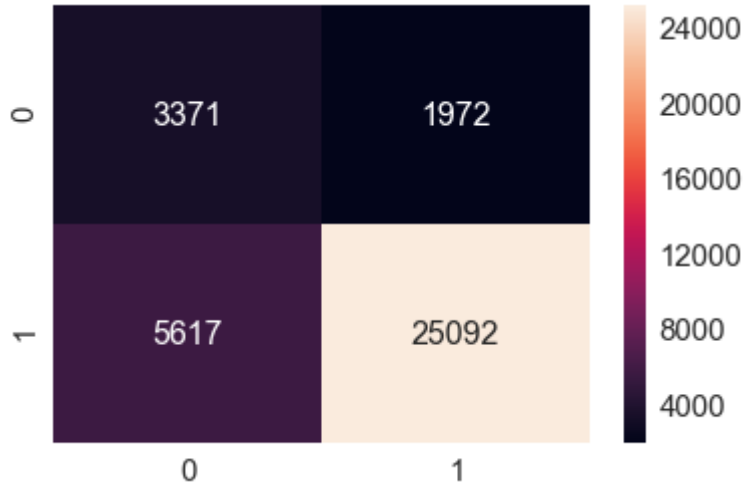
In [62]:

```
# Confusion Matrix Set1_test
nb.fit(set1_test,y_test)
y_test_pred_1 = nb.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)
sns.heatmap(conf_matr_df_test_1,annot=True,annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999124271144 for threshold -1.819
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17dc8320>
```



2.1.4 Top 10 important features of positive class from SET 1

In [174]:

```
nb.fit(set1_test,y_test)
pos_class_prob_sorted = nb.feature_log_prob_[0, :].argsort()
print(np.take(vectorizer.get_feature_names(), pos_class_prob_sorted[-10::],axis=None, o
```

```
['lifelong' 'magical' 'fantastic' 'emotional' 'another' 'staying'
 'cooking' 'level' 'current' 'opportunity']
```

2.1.5 Top 10 important features of negative class from SET 1

In [173]:

```
nb.fit(set1_test,y_test)
neg_class_prob_sorted = nb.feature_log_prob_[1, :].argsort()
print(np.take(vectorizer.get_feature_names(), neg_class_prob_sorted[10:],axis=None, out
```

```
['bands' 'go' 'comfort' ... 'level' 'current' 'opportunity']
```

2.2 Applying Naive Bayes on TFIDF, SET 2

Merging features encoding numerical + categorical features TFIDF, SET 2

In [64]:

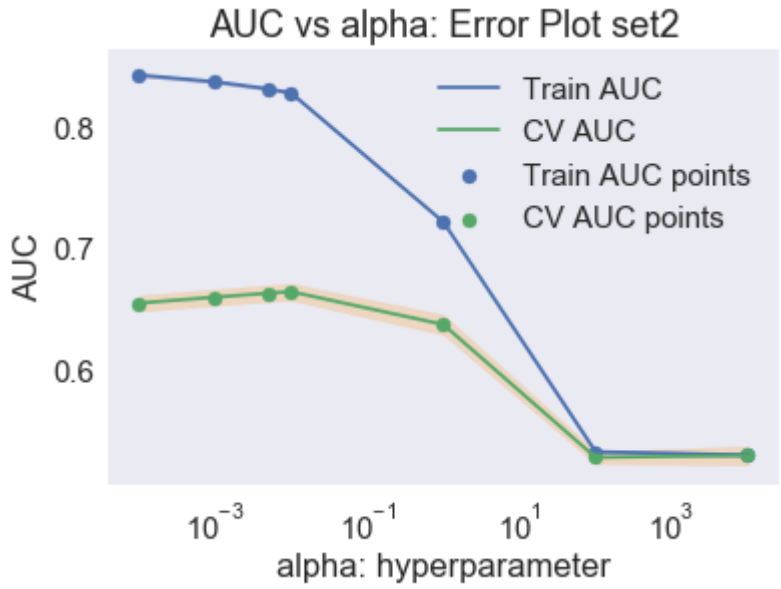
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set2_ = all categorical features + numarical features + essays_tfidf + project_title_

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, 14796) (49041,)
shape of set2_test and y_test   : (36052, 14796) (36052,)
shape of set2_cv and y_cv       : (24155, 14796) (24155,)
```

2.2.1 Hyper parameter Tuning to find best alpha:: GridSearchcv

In [65]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
nb = MultinomialNB(class_prior=[0.5,0.5])
p = [0.0001, 0.001, 0.005, 0.01, 1, 100, 10000]
params = {'alpha':[0.0001, 0.001, 0.005, 0.01, 1, 100, 10000]}
clf = GridSearchCV(nb, 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['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['alpha'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

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

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

plt.legend()
plt.xscale('log')
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs alpha: Error Plot set2")
plt.grid()
plt.show()
```

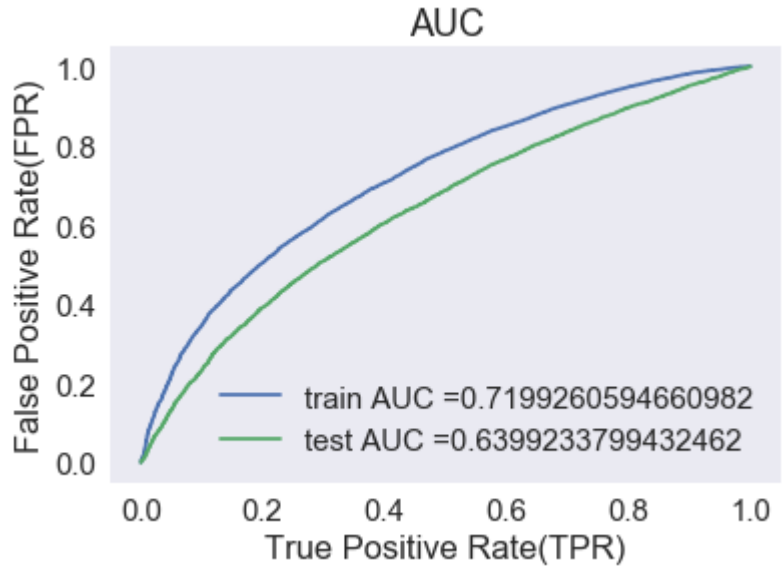


Wall time: 14.9 s

2.2.2 Train model using the best hyper-parameter alpha value set2

In [66]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_curve, auc
nb = MultinomialNB(alpha=1, class_prior=[0.5,0.5])
nb.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, nb.predict_log_proba(set2_train)[
test_fpr, test_tpr, thresholds = roc_curve(y_test, nb.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()
```




```
Wall time: 699 ms
```

2.2.3 Confustion Matrix Set2_train

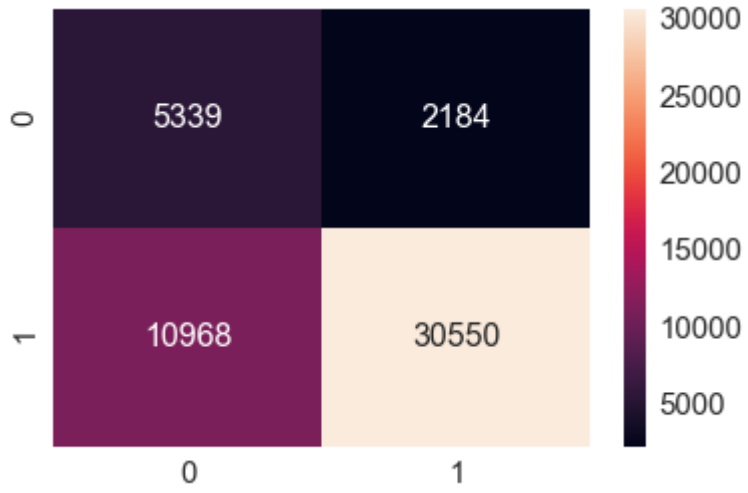
In [67]:

```
# Confustion Matrix Set2_train
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred,thresl
train_fpr, train_fpr)), range(2),range(2))

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

```
the maximum value of tpr*(1-fpr) 0.24999999558268995 for threshold -0.875
```

Out[67]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17e1d780>
```



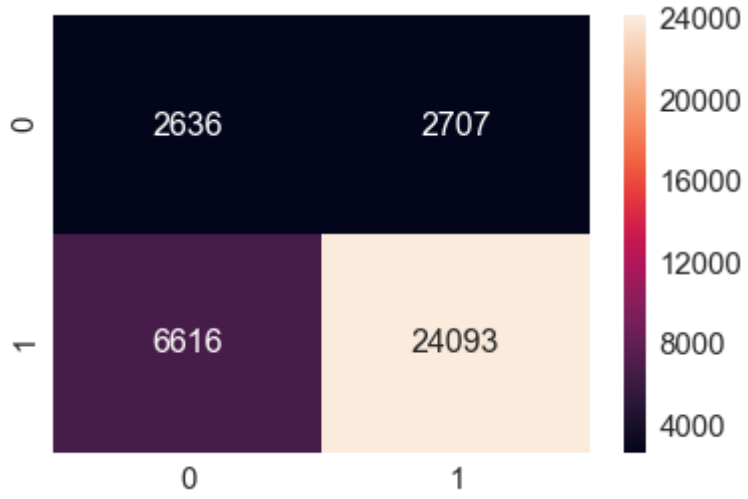
In [68]:

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

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) 0.24999999124271144 for threshold -0.579
```

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x27f44390>
```



2.2.4 Top 10 important features of positive class from SET 2

In [175]:

```
nb.fit(set2_test,y_test)
pos_class_prob_sorted = nb.feature_log_prob_[1, :].argsort()
print(np.take(vectorizer.get_feature_names(), pos_class_prob_sorted[-10::],axis=None, o
```

```
['2017' 'level' 'bag' 'current' 'opportunity' 'again' 'age' 'ahead' '21st'
 '2nd']
```

2.2.5 Top 10 important features of negative class from SET 2

In [166]:

```
nb.fit(set2_test,y_test)
neg_class_prob_sorted = nb.feature_log_prob_[0, :].argsort()
print(np.take(vectorizer.get_feature_names(), neg_class_prob_sorted[:10],axis=None, out
```

```
['flexible' 'firsties' 'cases' 'gifted' 'tiny' 'girls' 'time' 'career'
 'care' 'thousand']
```

3. Conclusions

In [69]:

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

p.field_names = ["Vectorizer", "Model", "Hyper parameter(alpha)", "AUC"]

p.add_row(["BOW","MultinomialNB",1, 0.70])
p.add_row(["TFIDF","MultinomialNB", 1, 0.65])

print(p)
```

```
+------------+---------------+-------------------------+------+
| Vectorizer |     Model     | Hyper parameter(alpha) | AUC  |
+------------+---------------+-------------------------+------+
|    BOW     | MultinomialNB |           1             | 0.7  |
|   TFIDF    | MultinomialNB |           1             | 0.65 |
+------------+---------------+-------------------------+------+
```

Thank You.

Sign Off RAMESH BATTU