DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

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

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
```

```
import 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]:

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

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

Out[4]:

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

1.2 preprocessing of `project_subject_categories`

In [5]:

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

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:

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

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

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

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
```

```
    my_counter.update(word.split())

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

count of words in project_title

In [7]:

```
title_word_count=[]
```

In [8]:

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

In [24]:

```
project_data["title_word_count"]=title_word_count
project_data["project_grade_category"]=project_data["project_grade_category"].str.replace(" ","")
```

In [25]:

```
project_data["project_grade_category"]=project_data["project_grade_category"].str.replace("-","")
```

In [26]:

```
project_data["project_grade_category"].head(5)
```

Out[26]:

```
0    GradesPreK2
1       Grades68
2       Grades68
3    GradesPreK2
4    GradesPreK2
Name: project_grade_category, dtype: object
```

1.3 Text preprocessing

In [27]:

```
# 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 [28]:

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

Out[28]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	GradesP

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grac

count of no.of words in essay

In [29]:

```
essay_word_count=[]
```

In [30]:

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

In [31]:

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

In [32]:

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

Out[32]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	GradesF
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grac
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grac
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	GradesF
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	GradesF

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

In [33]:

```
#train and test data splitting
from sklearn.model_selection import train_test_split
```

```
test_size=0.33,stratify=project_data["project_is_approved"])
```

In [34]:

```
x_train,x_cv,y_train,y_cv=train_test_split(x_train,y_train,test_size=0.33,stratify=y_train)
```

In [35]:

```
x_train.drop(["project_is_approved"],axis=1,inplace=True)
x_test.drop(["project_is_approved"],axis=1,inplace=True)
x_cv.drop(["project_is_approved"],axis=1,inplace=True)
```

In [20]:

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

Text preprocessing

In [36]:

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

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
```

allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still.nannan
==
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n \r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage r learners; they are like sponges, absorbing all the information and experiences and keep on wanti ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic tures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project t o make our new school year a very successful one. Thank you!nannan
==
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced pr ice lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l earn through games, my kids don't want to sit and do worksheets. They want to learn to count by ju mping and playing. Physical engagement is the key to our success. The number toss and color and sh ape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan
==
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The grea t teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% Af rican-American, making up the largest segment of the student body. A typical school in Dallas is m ade up of 23.2% African-American students. Most of the students are on free or reduced lunch. We a ren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smar t, effective, efficient, and disciplined students with good character.In our classroom we can util ize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the so und enough to receive the message. Due to the volume of my speaker my students can't hear videos o r books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my s tudents will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will all ow me to have more room for storage of things that are needed for the day and has an extra part to it I can use. The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.nannan
==

In [37]:

```
# 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)
```

In [38]:

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

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

In [39]:

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

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

In [40]:

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

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

In [41]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
```

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

Preprcessing of Train data

In [42]:

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

```
100%|██████████| 49041/49041 [11
:13<00:00, 72.80it/s]
```

In [43]:

```
# after preprocesing
preprocessed_essays_train[1000]
```

Out[43]:

```
'the majority students not enjoy reading doing well school not philosophy cool district times i st
ruggle keep students engaged i lose lot best readers writers drug use school activity seems take p
recedent time i able connect world i begin hook new readers while students may not love reading cl
assics enjoy dystopian science fiction stories they love discussing government meaning life places
world dystopian literature helped conversations students connect conversation works literature two
years ago i created english elective school called the dawn new genre history science fiction birt
h dystopia my class capacity interest carried throughout following year i happy report i teaching
elective upcoming 2016 17 school year unfortunately school not budget new books i able get donatio
n class set brave new world books two years ago i hoping expose students different literary genres
1984 would excellent addition classroom lives the dystopian genre flourishing young adult category
i want students ask authors writing apocalyptic big brother centered novels geared age group this
says something culture direction literature moving towards novels like help spark conscious
thought get younger generations engaged independent reading endeavors before ask questions must lo
ok back origin genre nannan'
```

Out[44]:

```
0    GradesPreK2
1       Grades68
2       Grades68
3    GradesPreK2
Name: project_grade_category, dtype: object
```

preprocessing of test data

In [45]:

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

```
100%|| 36052/36052 [05:41<00:00, 105.72it/s]
```

In [46]:

```
# after preprocesing
preprocessed_essays_test[1000]
```

Out[46]:

```
'our middle school outer edge large urban suburban district diverse population ranging affluent im
poverished we stem pbl project based learning focused school students love creative showing know w
e great elementary schools feeding one middle school kids generally familiar idea project based le
arning start middle school 6th grade spent focusing refining knowledge developing deeper content u
nderstanding our greatest challenge broad range student access resources created large diverse sch
ool cluster while students eager learn hands student driven projects not students equal access res
ources allow participate projects fullest all students love technology engage quickly digital acti
vities not students afford device often classroom feels divided haves nots due difference students
love participate digital labs using tablets we use tablets open new world science resources labs s
tudents allowing experience many hands experiences science one device we additionally use tablets
create digital data portfolios students class learn growth mindset set yearly semester monthly wee
kly goals track progress toward goals this currently expensive paper heavy progress by funding
tablets allow us move important work digital realm practice preach regarding conservation
resources still teaching integral skills goal setting self monitoring action plans goal attainment
nannan'
```

Preprocessing of Cross-validation data

In [47]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_cv = []
# tqdm is for printing the status bar
for sentance in tqdm(x_cv['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
```

```
    preprocessed_essays_cv.append(sent.lower().strip())
```

```
100%|| 24155/24155 [04
:28<00:00, 90.11it/s]
```

In [48]:

```
# after preprocesing
preprocessed_essays_cv[1000]
```

Out[48]:

```
'i proud bilingual teacher north session bronx my students faced several challenges classroom i
looking provide students creative meaningful learning experiences order successful life learning c
areer i teach low income school district many students raised single parent households receive fre
e lunch based socioeconomic status these obstacles may prevent getting ahead early life may not
provide life experiences many us see normal my focus scaffold potential growth i may not able cont
rol home lives however i certainly make difference school day control provide facilitate creative
meaningful learning experience students by teaching creative positive way i inspiring earliest lea
rners continue path academic excellence motivation one important factors determining successful se
cond language acquisition the used television i pads classroom not support english language learne
rs listening speaking skills motivation well by watching appropriate instructional shows between l
ion dora the explorer maya miguel postcards from buster students opportunity learn new words impro
ve literacy english language skills value cultural diversity your generous donation project not im
prove kindergarteners listening speaking reading writing skills also give plentiful opportunities
review activities rehearse language learning process entertaining enjoyable appropriate
instructional programs apps cameras foster literacy skills the use camera help zoom small items se
e describe details also show tell activities science experiments by watching educational show maya
miguel students opportunity build awareness appreciation many cultures america it encourage
students value respect better understand variety cultures perspectives traditions languages
experiences the implementation i pads classroom help students build understanding english language
special emphasis vocabulary my students also develop rehearse phonemic awareness letters sound rec
ognition develop positive attitude toward knowing learning one language nannan'
```

1.4 Preprocessing of `project_title`

In [33]:

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

In [49]:

```
print(project_data['project_title'].values[2000])
print("="*50)
print(project_data['project_title'].values[1500])
print("="*50)
print(project_data['project_title'].values[10000])
print("="*50)
print(project_data['project_title'].values[12000])
print("="*50)
print(project_data['project_title'].values[1699])
print("="*50)
```

```
Steady Stools for Active Learning
==================================================
Listening Center
==================================================
Family Book Clubs
==================================================
Cooperative Learning in 4th Grade!
==================================================
Standing Options for Energetic Learners
==================================================
```

preproccessing of project_title feature for train data

```
preprocessed_titles_train = []

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

```
100%|| 49041/49041 [00:19<00:00, 2551.27it/s]
```

In [51]:

```
preprocessed_titles_train[1000]
```

Out[51]:

```
'a dystopian future through eyes george orwell'
```

preproccessing of project_title feature for test data

In [52]:

```
preprocessed_titles_test = []

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

```
100%|| 36052/36052 [00:14<00:00, 2548.91it/s]
```

In [53]:

```
preprocessed_titles_test[1000]
```

Out[53]:

```
'opening world science labs'
```

preproccessing of project_title feature for cross validation data

In [54]:

```
preprocessed_titles_cv = []

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

In [55]:

```
preprocessed_titles_cv[1000]
```

Out[55]:

```
'strengthening language acquisition instructional technology'
```

1.5 Preparing data for models

In [56]:

```
project_data.columns
```

Out[56]:

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

we are going to consider

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

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

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

2.2 Make Data Model Ready: encoding Numerical and categorical data

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

Vectorizing categorical data

One hot encoding of clean_categories

In [57]:

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

from sklearn.feature_extraction.text import CountVectorizer

vectorizer_proj = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary
=True)
vectorizer_proj.fit(x_train['clean_categories'].values)
```

```
categories_one_hot_cv = vectorizer_proj.transform(x_cv['clean_categories'].values)

print(vectorizer_proj.get_feature_names())

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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix of Train data after one hot encoding  (49041, 9)
Shape of matrix of Test data after one hot encoding  (36052, 9)
Shape of matrix of CV data after one hot encoding  (24155, 9)
```

One hot encoding of clean_subcategories

In [58]:

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

vectorizer_sub_proj = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False
, binary=True)
vectorizer_sub_proj.fit(x_train['clean_subcategories'].values)

sub_categories_one_hot_train = vectorizer_sub_proj.transform(x_train['clean_subcategories'].values
)
sub_categories_one_hot_test = vectorizer_sub_proj.transform(x_test['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer_sub_proj.transform(x_cv['clean_subcategories'].values)


print(vectorizer_sub_proj.get_feature_names())

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix of Train data after one hot encoding  (49041, 30)
Shape of matrix of Test data after one hot encoding  (36052, 30)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 30)
```

One hot encoding of school_state

In [59]:

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

In [60]:

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

In [61]:

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

vectorizer_states = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()),
```

```
school_state_categories_one_hot_train = vectorizer_states.transform(x_train['school_state'].values
)
school_state_categories_one_hot_test = vectorizer_states.transform(x_test['school_state'].values)
school_state_categories_one_hot_cv = vectorizer_states.transform(x_cv['school_state'].values)

print(vectorizer_states.get_feature_names())

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

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'I
A', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ',
'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX
', 'CA']
Shape of matrix of Train data after one hot encoding  (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 51)
```

One hot encoding of project_grade_category

In [62]:

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

In [63]:

```
project_grade_cat_dict = dict(my_counter)
```

In [64]:

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

In [65]:

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

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()),
lowercase=False, binary=True)
vectorizer_grade.fit(x_train['project_grade_category'].values)

project_grade_categories_one_hot_train =
vectorizer_grade.transform(x_train['project_grade_category'].values)
project_grade_categories_one_hot_test = vectorizer_grade.transform(x_test['project_grade_category'
].values)
project_grade_categories_one_hot_cv = vectorizer_grade.transform(x_cv['project_grade_category'].va
lues)

print(vectorizer_grade.get_feature_names())

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

```
['Grades912', 'Grades68', 'Grades35', 'GradesPreK2']
Shape of matrix of Train data after one hot encoding  (49041, 4)
Shape of matrix of Test data after one hot encoding  (36052, 4)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 4)
```

One hot encoding of teacher_prefix

In [66]:

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

In [67]:

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

In [68]:

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

vectorizer_teacher = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys()), lower
case=False, binary=True)
vectorizer_teacher.fit(x_train['teacher_prefix'].values.astype("U"))

teacher_prefix_categories_one_hot_train = vectorizer_teacher.transform(x_train['teacher_prefix'].v
alues.astype("U"))
teacher_prefix_categories_one_hot_test =
vectorizer_teacher.transform(x_test['teacher_prefix'].values.astype("U"))
teacher_prefix_categories_one_hot_cv = vectorizer_teacher.transform(x_cv['teacher_prefix'].values.
astype("U"))

print(vectorizer_teacher.get_feature_names())

print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_test.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_cv.shape)
```

```
['nan', 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encoding  (49041, 6)
Shape of matrix after one hot encoding  (36052, 6)
Shape of matrix after one hot encoding  (24155, 6)
```

In [69]:

```
# 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, binary=True
)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [70]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=
True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [55]:

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

2.3 Make Data Model Ready: encoding eassay, and project_title

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

Train,Test and Cross Validation data on essays

In [71]:

```
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(preprocessed_essays_train)
text_bow_train = vectorizer_bow_essay.transform(preprocessed_essays_train)
print("Shape of matrix after one hot encodig ",text_bow_train.shape)
```

```
Shape of matrix after one hot encodig  (49041, 12211)
```

In [72]:

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

```
Shape of matrix after one hot encoding  (36052, 12211)
```

In [73]:

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

```
Shape of matrix after one hot encoding  (24155, 12211)
```

Train,Test and Cross Validation data on Titles

In [74]:

```
vectorizer_bow_title=CountVectorizer(min_df=10)
vectorizer_bow_title.fit(preprocessed_titles_train)
title_bow_train = vectorizer_bow_title.transform(preprocessed_titles_train)
print("Shape of matrix after one hot encoding ",title_bow_train.shape)
```

```
Shape of matrix after one hot encoding  (49041, 2089)
```

In [75]:

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

In [76]:

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

```
Shape of matrix after one hot encoding  (24155, 2089)
```

In [77]:

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

1.5.2.2 TFIDF vectorizer

Train,Test,Cross validation Data on Essays

In [78]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(preprocessed_essays_train)

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

```
Shape of matrix after one hot encoding  (49041, 12211)
```

In [79]:

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

```
Shape of matrix after one hot encoding  (36052, 12211)
```

In [80]:

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

```
Shape of matrix after one hot encoding  (24155, 12211)
```

Train,Test,Cross validation Data on titles

In [81]:

```
vectorizer_tfidf_titles = TfidfVectorizer(min_df=10)

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

```
Shape of matrix after one hot encoding  (49041, 2089)
```

In [82]:

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

In [83]:

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

```
Shape of matrix after one hot encoding  (24155, 2089)
```

Vectorizing numerical features

In [84]:

```
#PRICE

# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[84]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [85]:

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

In [86]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

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

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

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

In [87]:

```
normalizer = Normalizer()

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

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

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

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

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

In [88]:

```
#Teacher_number_of_previously_posted_projects

normalizer = Normalizer()

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

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

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

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

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

In [89]:

```
#Title word count

normalizer = Normalizer()

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

```
title_word_count_test = normalizer.transform(X_test['title_word_count'].values.reshape(-1,1))

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

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

In [90]:

```
#Essay wrod count

normalizer_ess_count = Normalizer()

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

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

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

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

Assignment 4: 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 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 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

- Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

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

2. Naive Bayes

2.4 Appling NB() on different kind of featurization as mentioned in the instructions

Apply Naive Bayes on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

2.4.1 Applying Naive Bayes on BOW, SET 1

In [0]:

```
# Please write all the code with proper documentation
```

In [91]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
title_bow_train,
               text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test, quantity_test,
               prev_projects_test, title_word_count_test, essay_word_count_test, title_bow_test, te
xt_bow_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv,
quantity_cv,
               prev_projects_cv, title_word_count_cv, essay_word_count_cv, title_bow_cv,
text_bow_cv)).tocsr()
```

In [92]:

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

```
Final Data matrix
(49041, 14405) (49041,)
(24155, 14405) (24155,)
(36052, 14405) (36052,)
====================================================================================================
```

value

In [93]:

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

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

    return y_data_pred
```

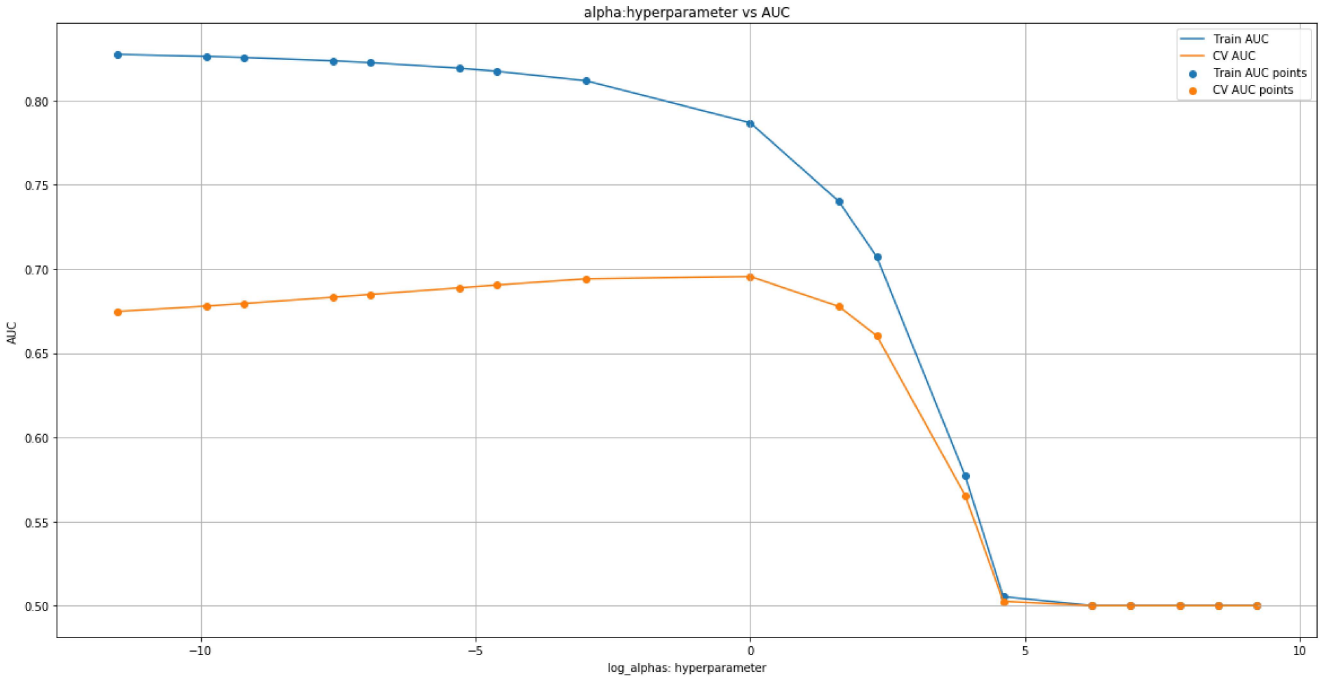
Random alpha values

In [94]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

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

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

"""

train_auc = []
cv_auc = []
log_alphas=[]

alphas= [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 1, 5,10,50,100,500,1000,2500,5
000,10000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha=i)
    nb.fit(X_tr, y_train)

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


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
for a in tqdm(alphas):
    b=math.log(a)
    log_alphas.append(b)
plt.figure(figsize=(20,10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')
```

```
plt.ylabel("AUC")
plt.title("alpha:hyperparameter vs AUC")
plt.grid()
plt.show()
```

```
100%|██████████| 18/18
[00:26<00:00,  1.71s/it]
100%|██████████| 18/18
[00:00<00:00, 750.32it/s]
```



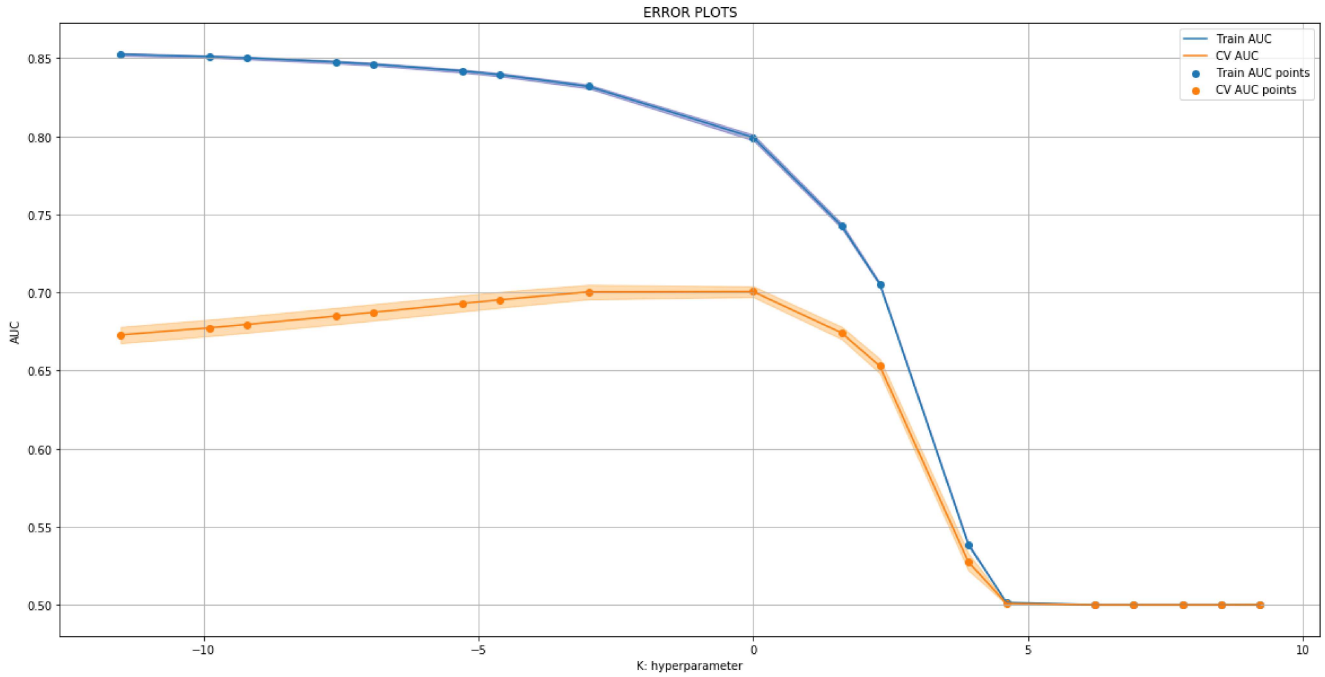
Gridsearch

In [95]:

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

nb=MultinomialNB()

parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 1, 5,10,50,100,5
00,1000,2500,5000,10000]}

clf = GridSearchCV(nb, parameters, cv= 5, scoring='roc_auc')

clf.fit(X_tr, 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']

log_alphas=[]

alphas= [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 1, 5,10,50,100,500,1000,2500,5
000,10000]

for a in tqdm(alphas):
    b=math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(20,10))
plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas,train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.3,col
or='darkblue')
```

```
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkoran
ge')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

```
100%|██████████| 18/18
[00:00<00:00, 4487.22it/s]
```



In [96]:

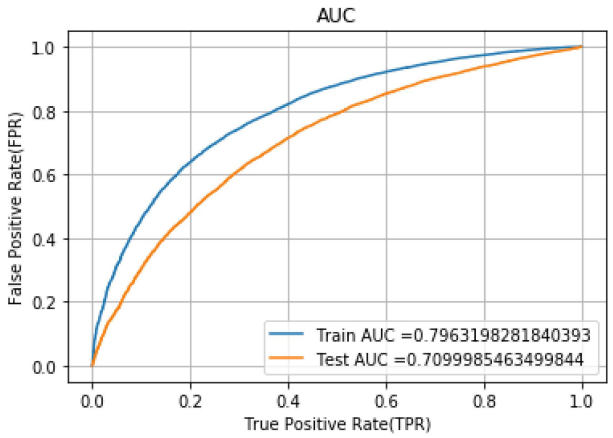
```
best_k_1=0.5
```

Train model using the best hyper-parameter value

In [97]:

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

from sklearn.metrics import roc_curve, auc

nb=MultinomialNB(alpha=best_k_1)
nb.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(nb, X_tr)
y_test_pred = batch_predict(nb, X_te)

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
```

```
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



confusion matrix

In [99]:

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

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

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

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

Train data

In [100]:

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

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999818661462 for threshold 0.115
[[ 3712  3714]
 [ 5002 36613]]
```

In [101]:

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

```
the maximum value of tpr*(1-fpr) 0.2499999818661462 for threshold 0.115
```

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

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x217bcf7b198>
```

Test data

In [103]:

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

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

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.577
[[ 2878  2581]
 [ 6944 23649]]
```

In [104]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, test_f
pr, test_fpr)), range(2),
                                 range(2))
```

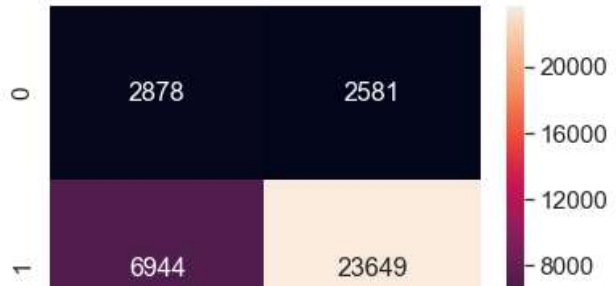
```
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.577
```

In [105]:

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

Out[105]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21780ebbbe0>
```



2.4.1.1 Top 10 important features of positive class from SET 1

In [91]:

```
# Please write all the code with proper documentation
```

In [106]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
title_bow_train,
               text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test, quantity_test,
               prev_projects_test, title_word_count_test, essay_word_count_test, title_bow_test, te
xt_bow_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv,
quantity_cv,
               prev_projects_cv, title_word_count_cv, essay_word_count_cv, title_bow_cv,
text_bow_cv)).tocsr()
```

In [107]:

```
nb_bow=MultinomialNB(alpha=0.5,class_prior = [0.5, 0.5])
nb_bow.fit(X_tr,y_train)
```

Out[107]:

```
MultinomialNB(alpha=0.5, class_prior=[0.5, 0.5], fit_prior=True)
```

In [108]:

```
bow_feature_probs={}

for a in range(14244):
    bow_feature_probs[a]=int(nb_bow.feature_log_prob_[0,a])
```

In [109]:

```
len(bow_feature_probs.values())
```

Out[109]:

```
14244
```

In [ ]:

```

```

In [110]:

```
bow_feature_names=[]
```

```
bow_feature_names.extend(vectorizer_proj.get_feature_names())
```

In [112]:

```
bow_feature_names.extend(vectorizer_sub_proj.get_feature_names())
```

In [113]:

```
bow_feature_names.extend(vectorizer_states.get_feature_names())
```

In [114]:

```
bow_feature_names.extend(vectorizer_teacher.get_feature_names())
```

In [115]:

```
bow_feature_names.extend(vectorizer_grade.get_feature_names())
```

In [116]:

```
len(bow_feature_names)
```

Out[116]:

100

In [117]:

```
bow_feature_names.append("price")
```

In [118]:

```
bow_feature_names.append("quantity")
```

In [119]:

```
bow_feature_names.append("prev_proposed_projects")
```

In [120]:

```
bow_feature_names.append("title_word_count")
```

In [121]:

```
bow_feature_names.append("essay_word_count")
```

In [122]:

```
len(bow_feature_names)
```

Out[122]:

105

In [123]:

```
bow_feature_names.extend(vectorizer_bow_title.get_feature_names())
```

In [124]:

```
len(bow_feature_names)
```

Out[124]:

```
2194
```

In [125]:

```
bow_feature_names.extend(vectorizer_bow_essay.get_feature_names())
```

In [126]:

```
len(bow_feature_names)
```

Out[126]:

```
14405
```

In [ ]:

```

```

In [127]:

```
bow_features_probs_positive= nb_bow.feature_log_prob_[1,:]
bow_features_probs_positive.view()
```

Out[127]:

```
array([ -9.3799895 ,  -9.3799895 ,  -7.99347412, ..., -13.36632454,
       -13.19197115, -12.91433942])
```

In [128]:

```
final_bow_features_pos = pd.DataFrame({'feature_prob_estimates' : bow_features_probs_positive,
                                       'feature_names' : bow_feature_names})
final_bow_features_pos.head(5)
```

Out[128]:

	feature_prob_estimates	feature_names
0	-9.379990	Warmth
1	-9.379990	Care_Hunger
2	-7.993474	History_Civics
3	-7.439145	Music_Arts
4	-7.283562	AppliedLearning

In [129]:

```
b = final_bow_features_pos.sort_values(by=['feature_prob_estimates'],ascending = True)
type(b)
```

Out[129]:

```
pandas.core.frame.DataFrame
```

In [130]:

```
b.head(25)
```

Out[130]:

	feature_prob_estimates	feature_names
95	-16.410847	Mrs.

	feature_prob_estimates	feature_names
99	-16.410847	GradesPreK2
98	-16.410847	Grades35
94	-14.464937	Ms.
9581	-14.213622	nudge
7665	-14.213622	im
2518	-14.213622	acronym
5938	-14.012952	employing
268	-14.012952	beginnings
5623	-14.012952	downward
12893	-14.012952	swag
1260	-14.012952	manipulative
1298	-14.012952	mental
1590	-14.012952	public
753	-14.012952	evidence
329	-14.012952	breathe
422	-14.012952	chefs
1657	-13.845898	results
4866	-13.845898	creat
8275	-13.845898	kale
1635	-13.845898	redesign
8570	-13.845898	lifeskills
13795	-13.845898	useable
1866	-13.845898	stimulating

In [167]:

```
des = final_bow_features_pos.sort_values(by=['feature_prob_estimates'],ascending = False)
type(des)
des.head(10)
```

Out[167]:

	feature_prob_estimates	feature_names
12716	-3.091298	students
11722	-4.234414	school
9349	-4.549020	my
8490	-4.601609	learning
4248	-4.626925	classroom
13125	-4.852728	the
9543	-4.887526	not
13159	-4.891770	they
8486	-4.942616	learn
7381	-4.967625	help

Negative features from bow model

2.4.1.2 Top 10 important features of negative class from SET 1

In [117]:

```
# Please write all the code with proper documentation
```

In [131]:

```
bow_features_probs_negative= nb_bow.feature_log_prob_[0,:]
len(bow_features_probs_negative)
```

Out[131]:

```
14405
```

In [132]:

```
final_bow_features_neg = pd.DataFrame({'feature_prob_estimates' : bow_features_probs_negative,
                                       'feature_names' : bow_feature_names})
final_bow_features_neg.head(5)
```

Out[132]:

	feature_prob_estimates	feature_names
0	-9.964124	Warmth
1	-9.964124	Care_Hunger
2	-8.020888	History_Civics
3	-7.376430	Music_Arts
4	-7.082618	AppliedLearning

In [133]:

```
a = final_bow_features_neg.sort_values(by=['feature_prob_estimates'],ascending = True)
type(a)
```

Out[133]:

```
pandas.core.frame.DataFrame
```

In [134]:

```
a.head(25)
```

Out[134]:

	feature_prob_estimates	feature_names
14043	-14.636953	warranty
9861	-14.636953	oversee
5088	-14.636953	decks
12921	-14.636953	symbolism
5778	-14.636953	eclipse
3176	-14.636953	autobiographies
5089	-14.636953	declaration
9865	-14.636953	overtime
859	-14.636953	forever
2206	-14.636953	115
5082	-14.636953	decimal
1064	-14.636953	informed
13945	-14.636953	viruses
5093	-14.636953	decodable
9866	-14.636953	overview
11789	-14.636953	searches
11679	-14.636953	saxophone

	feature_prob_estimates	feature_names
3157	-14.636953	auggie
10951	-14.636953	raring
8770	-14.636953	madness
526	-14.636953	could
11405	-14.636953	revamp
7976	-14.636953	instrumentation
5075	-14.636953	decals

In [168]:

```
desc = final_bow_features_neg.sort_values(by=['feature_prob_estimates'],ascending = False)
type(desc)
desc.head(10)
```

Out[168]:

	feature_prob_estimates	feature_names
12716	-3.111101	students
11722	-4.188731	school
8490	-4.520898	learning
9349	-4.578729	my
4248	-4.693428	classroom
9543	-4.874511	not
8486	-4.875432	learn
13159	-4.892285	they
7381	-4.913370	help
13125	-4.934175	the

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [122]:

```
# Please write all the code with proper documentation
```

In [135]:

```
X_tr = hstack((categories_one_hot_train,
sub_categories_one_hot_train,school_state_categories_one_hot_train,
project_grade_categories_one_hot_train,teacher_prefix_categories_one_hot_train, price_train,
quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
text_tfidf_train, title_tfidf_train)).tocsr()
X_te = hstack((categories_one_hot_test,
sub_categories_one_hot_test,school_state_categories_one_hot_test,
project_grade_categories_one_hot_test,teacher_prefix_categories_one_hot_test, price_test,
quantity_test, prev_projects_test,title_word_count_test, essay_word_count_test, text_tfidf_test,
title_tfidf_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,school_state_categories_one_hot_cv
, project_grade_categories_one_hot_cv,teacher_prefix_categories_one_hot_cv, price_cv, quantity_cv,
prev_projects_cv, title_word_count_cv,essay_word_count_cv, text_tfidf_cv, title_tfidf_cv)).tocsr()
```

In [136]:

```
nb_tfidf = MultinomialNB(alpha = 0.1,class_prior = [0.5, 0.5])
nb_tfidf.fit(X_tr, y_train)
```

Out[136]:

```
MultinomialNB(alpha=0.1, class_prior=[0.5, 0.5], fit_prior=True)
```

```
tfidf_features_probs_neg = {}
for a in range(14244) :
    tfidf_features_probs_neg[a] = nb_tfidf.feature_log_prob_[0,a]
```

In [138]:

```
len(tfidf_features_probs_neg)
```

Out[138]:

14244

In [139]:

```
tfidf_feature_names = []
```

In [140]:

```
tfidf_feature_names.extend(vectorizer_proj.get_feature_names())
```

In [141]:

```
tfidf_feature_names.extend(vectorizer_sub_proj.get_feature_names())
```

In [142]:

```
tfidf_feature_names.extend(vectorizer_states.get_feature_names())
```

In [143]:

```
tfidf_feature_names.extend(vectorizer_grade.get_feature_names())
```

In [144]:

```
tfidf_feature_names.extend(vectorizer_teacher.get_feature_names())
```

In [145]:

```
len(tfidf_feature_names)
```

Out[145]:

100

In [146]:

```
tfidf_feature_names.append("price")
```

In [147]:

```
tfidf_feature_names.append("quantity")
```

In [148]:

```
tfidf_feature_names.append("prev_proposed_projects")
```

In [149]:

```
tfidf_feature_names.append("title_word_count")
```

In [150]:

In [151]:

```
tfidf_feature_names.extend(vectorizer_tfidf_titles.get_feature_names())
```

In [152]:

```
tfidf_feature_names.extend(vectorizer_tfidf_essay.get_feature_names())
```

In [153]:

```
len(tfidf_feature_names)
```

Out[153]:

```
14405
```

Negative features from tfidf model

In [154]:

```
tfidf_features_probs_negat= nb_tfidf.feature_log_prob_[0,:]
len(tfidf_features_probs_negat)
```

Out[154]:

```
14405
```

In [155]:

```
final_tfidf_features_neg = pd.DataFrame({'feature_prob_estimates' : tfidf_features_probs_negat,
                                         'feature_names' : tfidf_feature_names})
final_tfidf_features_neg.head(5)
```

Out[155]:

	feature_prob_estimates	feature_names
0	-7.956669	Warmth
1	-7.956669	Care_Hunger
2	-6.006999	History_Civics
3	-5.362032	Music_Arts
4	-5.068077	AppliedLearning

In [156]:

```
c = final_tfidf_features_neg.sort_values(by=['feature_prob_estimates'],ascending = True)
type(c)
```

Out[156]:

```
pandas.core.frame.DataFrame
```

In [157]:

```
c.head(25)
```

Out[157]:

	feature_prob_estimates	feature_names
1534	-14.231431	poverty

	feature_prob_estimates	feature_names
13734	-14.231431	unseen
12229	-14.231431	social
1716	-14.231431	season
5326	-14.231431	did
5333	-14.231431	diets
5334	-14.231431	differ
2557	-14.231431	addicted
9195	-14.231431	modernize
9193	-14.231431	moderately
13298	-14.231431	tomatoes
9236	-14.231431	monster
5387	-14.231431	diplomas
5398	-14.231431	dirt
2543	-14.231431	adage
2540	-14.231431	actual
723	-14.231431	enjoy
7772	-14.231431	inclusiveness
5408	-14.231431	disagreements
728	-14.231431	enthusiastic
2527	-14.231431	activates
13314	-14.231431	toolkit
732	-14.231431	equal
5437	-14.231431	discovering

In [170]:

```
c1 = final_tfidf_features_neg.sort_values(by=['feature_prob_estimates'],ascending = False)
type(c1)
c1.head(10)
```

Out[170]:

	feature_prob_estimates	feature_names
103	-3.016089	title_word_count
104	-3.016089	essay_word_count
101	-3.016089	quantity
100	-3.016089	price
102	-3.406247	prev_proposed_projects
8	-3.883866	Literacy_Language
93	-3.927457	GradesPreK2
7	-3.928798	Math_Science
92	-4.145163	Grades35
37	-4.361965	Mathematics

Positive features from tfidf model

In [158]:

```
tfidf_features_probs_positive= nb_tfidf.feature_log_prob_[1,:]
tfidf_features_probs_positive.view()
```

```
       -11.87866425, -11.07849954])
```

In [159]:

```
final_tfidf_features_pos = pd.DataFrame({'feature_prob_estimates' : tfidf_features_probs_positive,
                                         'feature_names' : tfidf_feature_names})
final_tfidf_features_pos.head(5)
```

Out[159]:

	feature_prob_estimates	feature_names
0	-7.319421	Warmth
1	-7.319421	Care_Hunger
2	-5.932375	History_Civics
3	-5.377970	Music_Arts
4	-5.222373	AppliedLearning

In [160]:

```
x = final_tfidf_features_pos.sort_values(by=['feature_prob_estimates'],ascending = True)
type(x)
```

Out[160]:

```
pandas.core.frame.DataFrame
```

In [161]:

```
x.head(25)
```

Out[161]:

	feature_prob_estimates	feature_names
95	-15.959009	Dr.
97	-15.959009	Mr.
98	-15.959009	Ms.
99	-15.959009	Mrs.
5576	-14.128507	domains
3534	-14.044352	boasts
429	-13.985831	children
7492	-13.913204	honestly
3849	-13.906448	camaraderie
10804	-13.803820	pumpkin
1028	-13.780983	ican
6080	-13.721195	environment
1002	-13.709860	history
6186	-13.686587	evidenced
1184	-13.673753	lesson
12073	-13.669442	simplicity
8138	-13.660343	iran
11706	-13.657050	scenery
1025	-13.653706	hungry
8270	-13.651386	jv
7106	-13.648052	grant
1172	-13.642116	learn

	feature_prob_estimates	feature_names
9579	-13.666866	fe[illegible]s
11596	-13.633370	rural

In [171]:

```
x1 = final_tfidf_features_pos.sort_values(by=['feature_prob_estimates'],ascending = False)
type(x1)
x1.head(10)
```

Out[171]:

	feature_prob_estimates	feature_names
100	-3.020206	price
103	-3.020206	title_word_count
101	-3.020206	quantity
104	-3.020206	essay_word_count
102	-3.328646	prev_proposed_projects
8	-3.741744	Literacy_Language
93	-3.928713	GradesPreK2
7	-4.002033	Math_Science
92	-4.089872	Grades35
38	-4.171327	Literacy

Conclusion

In [162]:

```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha:Hyper Parameter", "AUC"]
x.add_row(["BOW", "Naive Bayes", 0.5, 0.7])
x.add_row(["TFIDF", "Naive Bayes", 0.1, 0.67])
print(x)
```

```
+------------+-------------+-----------------------+------+
| Vectorizer |    Model    | Alpha:Hyper Parameter | AUC  |
+------------+-------------+-----------------------+------+
|    BOW     | Naive Bayes |          0.5          | 0.7  |
|   TFIDF    | Naive Bayes |          0.1          | 0.67 |
+------------+-------------+-----------------------+------+
```

In [163]:

```
y = PrettyTable()
y.field_names = ["Vectorizer", "Model", "K:Hyper Parameter", "AUC"]
y.add_row(["BOW", "KNN", 91, 0.63])
y.add_row(["TFIDF", "KNN", 85, 0.57])
print(y)
```

```
+------------+-------+--------------------+------+
| Vectorizer | Model | K:Hyper Parameter | AUC  |
+------------+-------+--------------------+------+
|    BOW     |  KNN  |         91         | 0.63 |
|   TFIDF    |  KNN  |         85         | 0.57 |
+------------+-------+--------------------+------+
```

Summary 1. Naive bayes seems to function better than KNN for both Bag of Words model (BOW) as well as Term Frequency Inverse