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	Descri
`project_id`	A unique identifier for the proposed project. **Example:** `p03`
`project_title`	Title of the project. **Exam** • `Art Will Make You Ha` • `First Grade`
`project_grade_category`	Grade level of students for which the project is targeted. One of the foll enumerated va • `Grades Pr` • `Grades` • `Grades` • `Grades`
`project_subject_categories`	One or more (comma-separated) subject categories for the project fro following enumerated list of va • `Applied Lear` • `Care & Hu` • `Health & Sp` • `History & Ci` • `Literacy & Lang` • `Math & Sci` • `Music & The` • `Special N` • `Wa` **Exam** • `Music & The` • `Literacy & Language, Math & Sci`
`school_state`	State where school is located (Two-letter U.S. postal (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_co **Example**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the pr **Exam** • `Lite` • `Literature & Writing, Social Scie`
`project_resource_summary`	An explanation of the resources needed for the project. **Exan** • `My students need hands on literacy materials to mar sensory ne`
`project_essay_1`	First application e
`project_essay_2`	Second application e
`project_essay_3`	Third application e
`project_essay_4`	Fourth application e

Feature	Descri
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04 12:43:56`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Exa** `bdf8baa8fedef6bfeec7ae4ff1c1`
`teacher_prefix`	Teacher's title. One of the following enumerated va • • • • • • `Teac`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same tea **Exampl**

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarnin
g: paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip insta
ll paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disable
d.  `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarnin
g: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'sc
hool_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://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tra
iling 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://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling 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/2289859
5/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.3 Text preprocessing

In [7]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [8]:
```
project_data.head(2)
```

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

In [9]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [10]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with 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 of your world.\"-Ludwig Wittgenstein Our English learner's have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English along side 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 home 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 English 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 educational 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 least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of 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 beautiful 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 the hard work put in during the school year, with a dunk tank being the most popular activity.My students 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 have 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 used 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 in 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 taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still.nannan

==========

How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting theme

d 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 and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging 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 pictures 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 to make our new school year a very successful one. Thank you!nannan
==========
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their 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 price 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 groove 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 develop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't 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 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 great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students are on free or reduced lunch. We aren'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 smart, effective, efficient, and disciplined students with good character.In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use. The table top chart has all of the letter, words and pictures for students to learn about different letters and it

```
is more accessible.nannan
==================================================
```

In [11]:
```python
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [12]:
```python
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations. \r\n\r\nThe materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or red
uced price lunch.  Despite their disabilities and limitations, my students lo
ve coming to school and come eager to learn and explore.Have you ever felt li
ke you had ants in your pants and you needed to groove and move as you were i
n a meeting? This is how my kids feel all the time. The want to be able to mo
ve as they learn or so they say.Wobble chairs are the answer and I love then
because they develop their core, which enhances gross motor and in Turn fine
motor skills. \r\nThey 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.
Physical 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.nannan
==================================================
```

In [13]:
```
# \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 lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations.     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 co
ming to school and come eager to learn and explore.Have you ever felt like yo
u had ants in your pants and you needed to groove and move as you were in a m
eeting? 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 becaus
e they develop their core, which enhances gross motor and in Turn fine motor
skills.   They also want to learn through games, my kids do not want to sit a
nd do worksheets. They want to learn to count by jumping and playing. Physica
l engagement is the key to our success. The number toss and color and shape m
ats can make that happen. My students will forget they are doing work and jus
t have the fun a 6 year old deserves.nannan
```

In [14]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays cognitive delays gross fine motor delays to autism They are eage
r beavers and always strive to work their hardest working past their limitati
ons 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 lun
ch Despite their disabilities and limitations my students love coming to scho
ol and come eager to learn and explore Have you ever felt like you had ants i
n your pants and you needed to groove 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 develop
their core 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 Physical 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 ol
d deserves nannan
```

In [15]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
, "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he'
, 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "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', 'becau
se', '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', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [16]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|
| 109248/109248 [01:00<00:00, 1811.97it/s]
```

In [17]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[17]:
```
'my kindergarten students varied disabilities ranging speech language delays
cognitive delays gross fine motor delays autism they eager beavers always str
ive work hardest working past limitations the materials ones i seek students
i teach title i school students receive free reduced price lunch despite disa
bilities limitations students love coming school come eager learn explore hav
e ever felt like ants pants needed groove move meeting this kids feel time th
e want able move learn say wobble chairs answer i love develop core enhances
gross motor turn fine motor skills they also want learn games kids not want s
it worksheets they want learn count jumping playing physical engagement key s
uccess the number toss color shape mats make happen my students forget work f
un 6 year old deserves nannan'
```

1.4 Preprocessing of `project_title`

In [18]:
```
# similarly you can preprocess the titles also
```

1.5 Preparing data for models

In [19]:
```
project_data.columns
```

Out[19]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

- 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 [20]:
```
# 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()), lowercas
e=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 [21]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowe
rcase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcateg
ories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducati
on', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geo
graphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Env
ironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'Spec
ialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [22]:
```
# you can do the similar thing with state, teacher_prefix and project_grade_ca
tegory also
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [23]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

In [24]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

In [25]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [26]:
```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/408403
9
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

# ============================

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

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

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

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


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

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

Out[26]:
```
'''\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/40
84039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n
f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in t
qdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        model[wo
rd] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return
model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ==================
==========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/
s]\nDone. 1917495  words loaded!\n\n# ============================\n\nwords =
[]\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in p
reproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in t
he coupus", len(words))\nwords = set(words)\nprint("the unique words in the c
oupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\np
rint("The number of words that are present in both glove vectors and our coup
us",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,
3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in wor
ds:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("wo
rd 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle
files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-v
ariables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as
f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [27]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how
-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [28]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|
| 109248/109248 [00:32<00:00, 3409.65it/s]

109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [29]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [30]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in th
is list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|
| 109248/109248 [03:50<00:00, 473.03it/s]

109248
300
```

In [31]:
```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

In [32]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'
}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [33]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/s
klearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mea
n and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_sc
alar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.resha
pe(-1, 1))
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

In [34]:
```
price_standardized
```

Out[34]:
```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

1.5.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [0]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [0]:
```
# 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 den
se matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standa
rdized))
X.shape
```

Out[0]: (109248, 16663)

In [0]:
```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the rea
der
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

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

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

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

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

   - 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 (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

2. Naive Baves

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
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_notebook as tqdm1
from tqdm import tqdm
import time
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

from sklearn.model_selection import train_test_split
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarnin
g: paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip insta
ll paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disable
d.  `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarnin
g: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [2]:
```
project_data = pd.read_csv('train_data.csv', nrows=50000)
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 (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'sc
hool_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]:
```
# not_accepted = project_data[project_data.project_is_approved==0]
# accepted = project_data[project_data.project_is_approved==1]
```

In [5]:
```
# print(accepted.shape)
# print(not_accepted.shape)
```

In [6]:
```
# # https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b
761be45a18
# # Upsampling minority class
# from sklearn.utils import resample
# not_accepted_upsampled = resample(not_accepted,
#                           replace=True, # sample with replacement
#                           n_samples=len(accepted), # match number in majorit
y class
#                           random_state=27) # reproducible results

# # combine majority and upsampled minority
# project_data = pd.concat([accepted, not_accepted_upsampled])

# # check new class counts
# print(project_data.project_is_approved.value_counts())
# print(project_data.shape)
```

In [ ]:

Text preprocessing(1)

In [7]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

In [8]:
```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(5)
```

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

In [9]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
my_counter
```

Out[9]:
```
Counter({'Literacy_Language': 23998,
         'History_Civics': 2689,
         'Health_Sports': 6538,
         'Math_Science': 18874,
         'SpecialNeeds': 6233,
         'AppliedLearning': 5569,
         'Music_Arts': 4699,
         'Warmth': 643,
         'Care_Hunger': 643})
```

In [10]:
```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))


# ind = np.arange(len(sorted_cat_dict))
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved category wise')
# plt.xticks(ind, list(sorted_cat_dict.keys()))
# plt.show()
# print(sorted_cat_dict)
```

In [11]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [12]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[12]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

In [13]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [14]:
```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))


# ind = np.arange(len(sorted_sub_cat_dict))
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_sub_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved state wise')
# plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
# plt.show()
```

In [15]:
```
# 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 [16]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-index
es-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'
}).reset_index()
price_data.head(2)
```

Out[16]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [17]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [18]:
```
#presence of the numerical digits in a strings with numeric : https://stackove
rflow.com/a/19859308/8089731
def hasNumbers(inputString):
    return any(i.isdigit() for i in inputString)
p1 = project_data[['id','project_resource_summary']]
p1 = pd.DataFrame(data=p1)
p1.columns = ['id','digits_in_summary']
p1['digits_in_summary'] =  p1['digits_in_summary'].map(hasNumbers)
# https://stackoverflow.com/a/17383325/8089731
p1['digits_in_summary'] = p1['digits_in_summary'].astype(int)
project_data = pd.merge(project_data, p1, on='id', how='left')
project_data.head(5)
```

Out[18]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

5 rows × 21 columns

Text preprocessing(2)

In [19]:
```
# 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 [20]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
, "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he'
, 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "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', 'becau
se', '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', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [21]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm1(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = re.sub('nannan', '', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

In [22]:
```
# preprocessed_essays
```

In [23]:
```
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for title in tqdm1(project_data['project_title'].values):
    _title = decontracted(title)
    _title = _title.replace('\\r', ' ')
    _title = _title.replace('\\"', ' ')
    _title = _title.replace('\\n', ' ')
    _title = re.sub('[^A-Za-z0-9]+', ' ', _title)
    # https://gist.github.com/sebleier/554280
    _title = ' '.join(e for e in _title.split() if e not in stopwords)
    preprocessed_titles.append(_title.lower().strip())
```

In [24]:
```
preprocessed_titles[1000]
```

Out[24]:
```
'sailing into super 4th grade year'
```

In [25]:
```
project_grade_catogories = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

project_grade_cat_list = []
for i in tqdm1(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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    project_grade_cat_list.append(temp.strip())
```

In [26]:
```
project_data['clean_project_grade_category'] = project_grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)
```

Out[26]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

2 rows × 21 columns

In [27]:
```
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','proj
ect_essay_4'], axis=1, inplace=True)
project_data.head(2)
```

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

In [28]:
```
#Replacing Nan's with maximum occured value: https://stackoverflow.com/a/51053
916/8089731
project_data['teacher_prefix'].value_counts().argmax()
project_data.fillna(value=project_data['teacher_prefix'].value_counts().argmax
(),axis=1,inplace=True)
```

In [29]:
```
project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_titles'] = preprocessed_titles
```

In [30]:
```
project_data.columns
```

Out[30]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantit
y',
       'digits_in_summary', 'clean_project_grade_category',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

In [31]:
```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debug
ging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the rea
der
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(project_data,project_data[
'project_is_approved'], test_size=0.33, stratify = project_data['project_is_ap
proved'])
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.
33, stratify=y_train)

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

1.4.1 Vectorizing Categorical data

In [33]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowe
rcase=False, binary=True)
vectorizer_cat.fit(X_train['clean_categories'].values)
print(vectorizer_cat.get_feature_names())


categories_one_hot_train = vectorizer_cat.transform(X_train['clean_categories'
].values)
categories_one_hot_cv = vectorizer_cat.transform(X_cv['clean_categories'].valu
es)
categories_one_hot_test = vectorizer_cat.transform(X_test['clean_categories'].
values)
print("Shape of matrix after one hot encodig_train ",categories_one_hot_train.
shape)
print("Shape of matrix after one hot encodig_cv ",categories_one_hot_cv.shape)
print("Shape of matrix after one hot encodig_test ",categories_one_hot_test.sh
ape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig_train  (22445, 9)
Shape of matrix after one hot encodig_cv  (11055, 9)
Shape of matrix after one hot encodig_test  (16500, 9)
```

In [34]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_sub_cat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys
()), lowercase=False, binary=True)
vectorizer_sub_cat.fit(X_train['clean_subcategories'].values)
print(vectorizer_sub_cat.get_feature_names())


sub_categories_one_hot_train = vectorizer_sub_cat.transform(X_train['clean_sub
categories'].values)
sub_categories_one_hot_cv = vectorizer_sub_cat.transform(X_cv['clean_subcatego
ries'].values)
sub_categories_one_hot_test = vectorizer_sub_cat.transform(X_test['clean_subca
tegories'].values)
print("Shape of matrix after one hot encodig_train ",sub_categories_one_hot_tr
ain.shape)
print("Shape of matrix after one hot encodig_cv ",sub_categories_one_hot_cv.sh
ape)
print("Shape of matrix after one hot encodig_test ",sub_categories_one_hot_tes
t.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducati
on', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geo
graphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Env
ironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'Spec
ialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig_train  (22445, 30)
Shape of matrix after one hot encodig_cv  (11055, 30)
Shape of matrix after one hot encodig_test  (16500, 30)
```

In [35]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_state = CountVectorizer( lowercase=False, binary=True)
vectorizer_state.fit(X_train['school_state'].values)
print(vectorizer_state.get_feature_names())


school_state_one_hot_train = vectorizer_state.transform(X_train['school_state'
].values)
school_state_one_hot_cv = vectorizer_state.transform(X_cv['school_state'].valu
es)
school_state_one_hot_test = vectorizer_state.transform(X_test['school_state'].
values)
print("Shape of matrix after one hot encodig_train ",school_state_one_hot_trai
n.shape)
print("Shape of matrix after one hot encodig_cv ",school_state_one_hot_cv.shap
e)
print("Shape of matrix after one hot encodig_test ",school_state_one_hot_test.
shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'I
A', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO',
'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR',
'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
Shape of matrix after one hot encodig_train  (22445, 51)
Shape of matrix after one hot encodig_cv  (11055, 51)
Shape of matrix after one hot encodig_test  (16500, 51)
```

In [36]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_teacherprefix = CountVectorizer( lowercase=False, binary=True)
vectorizer_teacherprefix.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer_teacherprefix.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
teacher_prefix_one_hot_train = vectorizer_teacherprefix.transform(X_train['tea
cher_prefix'].values.astype('U'))
teacher_prefix_one_hot_cv = vectorizer_teacherprefix.transform(X_cv['teacher_p
refix'].values.astype('U'))
teacher_prefix_one_hot_test = vectorizer_teacherprefix.transform(X_test['teach
er_prefix'].values.astype('U'))
print("Shape of matrix after one hot encodig_train ",teacher_prefix_one_hot_tr
ain.shape)
print("Shape of matrix after one hot encodig_cv ",teacher_prefix_one_hot_cv.sh
ape)
print("Shape of matrix after one hot encodig_test ",teacher_prefix_one_hot_tes
t[:5,:])
# print(X_train['teacher_prefix'].value_counts())
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig_train  (22445, 5)
Shape of matrix after one hot encodig_cv  (11055, 5)
Shape of matrix after one hot encodig_test    (0, 1)    1
  (1, 2)        1
  (2, 2)        1
  (3, 3)        1
  (4, 2)        1
```

In [37]:
```
print(project_data['clean_project_grade_category'].unique())
```

```
['GradesPreK-2' 'Grades6-8' 'Grades3-5' 'Grades9-12']
```

In [38]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
# https://stackoverflow.com/a/38161028/8089731
pattern = "(?u)\\b[\\w-]+\\b"
vectorizer_projectgrade = CountVectorizer(token_pattern=pattern, lowercase=Fal
se, binary=True)
vectorizer_projectgrade.fit(X_train['clean_project_grade_category'].values)
print(vectorizer_projectgrade.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
project_grade_category_one_hot_train = vectorizer_projectgrade.transform(X_tra
in['clean_project_grade_category'].values)
project_grade_category_one_hot_cv = vectorizer_projectgrade.transform(X_cv['cl
ean_project_grade_category'].values)
project_grade_category_one_hot_test = vectorizer_projectgrade.transform(X_test
['clean_project_grade_category'].values)
print("Shape of matrix after one hot encodig_train ",project_grade_category_on
e_hot_train.shape)
print("Shape of matrix after one hot encodig_cv ",project_grade_category_one_h
ot_cv.shape)
print("Shape of matrix after one hot encodig_test ",project_grade_category_one
_hot_test[:5,:])
```

```
['Grades3-5', 'Grades6-8', 'Grades9-12', 'GradesPreK-2']
Shape of matrix after one hot encodig_train  (22445, 4)
Shape of matrix after one hot encodig_cv  (11055, 4)
Shape of matrix after one hot encodig_test    (0, 0)    1
  (1, 0)        1
  (2, 0)        1
  (3, 3)        1
  (4, 3)        1
```

Vectorizing Numerical features

In [39]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/s
klearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and
standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_sc
alar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(X_train['price'].values.resh
ape(-1, 1))
price_standardized_cv = price_scalar.transform(X_cv['price'].values.reshape(-1
, 1))
price_standardized_test = price_scalar.transform(X_test['price'].values.reshap
e(-1, 1))
print(price_standardized_train.shape)
print(price_standardized_cv.shape)
print(price_standardized_test.shape)
```

```
Mean : 300.23218311427934, Standard deviation : 401.43554589763
(22445, 1)
(11055, 1)
(16500, 1)
```

In [40]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/s
klearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

quantity_scalar = StandardScaler()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the me
an and standard deviation of this data
# print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation : {np.sqrt(qua
ntity_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
quantity_standardized_train = quantity_scalar.transform(X_train['quantity'].va
lues.reshape(-1, 1))
quantity_standardized_cv = quantity_scalar.transform(X_cv['quantity'].values.r
eshape(-1, 1))
quantity_standardized_test = quantity_scalar.transform(X_test['quantity'].valu
es.reshape(-1, 1))
print(quantity_standardized_train.shape)
print(quantity_standardized_cv.shape)
print(quantity_standardized_test.shape)
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.


(22445, 1)
(11055, 1)
(16500, 1)
```

In [41]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/s
klearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

teacher_number_of_previously_posted_projects_scalar = StandardScaler()
teacher_number_of_previously_posted_projects_scalar.fit(X_train['teacher_numbe
r_of_previously_posted_projects'].values.reshape(-1,1)) # finding the mean and
standard deviation of this data
# print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_
[0]}, Standard deviation : {np.sqrt(teacher_number_of_previously_posted_projec
ts_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_train = teacher_numb
er_of_previously_posted_projects_scalar.transform(X_train['teacher_number_of_p
reviously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_standardized_cv = teacher_number_
of_previously_posted_projects_scalar.transform(X_cv['teacher_number_of_previou
sly_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_standardized_test = teacher_numbe
r_of_previously_posted_projects_scalar.transform(X_test['teacher_number_of_pre
viously_posted_projects'].values.reshape(-1, 1))
print(teacher_number_of_previously_posted_projects_standardized_train.shape)
print(teacher_number_of_previously_posted_projects_standardized_cv.shape)
print(teacher_number_of_previously_posted_projects_standardized_test.shape)
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.


(22445, 1)
(11055, 1)
(16500, 1)
```

In [42]:

```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debug
ging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the rea
der
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [43]:
```
X_train.head(2)
```

Out[43]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	prc
6025	178512	p148474	b8274a422e6a205748d98b23a3c134e9	Mrs.	NY	
26403	122685	p202047	df992790283e68a6338d9f9a86f14cb6	Mrs.	IL	

Bag of Words(BOW) on `project_TEXT/ESSAYS` (Train,Cv,Test)

In [44]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer_bow_essays = CountVectorizer(min_df=10)
vectorizer_bow_essays.fit(X_train['preprocessed_essays'])

text_bow_train = vectorizer_bow_essays.transform(X_train['preprocessed_essays'
])
text_bow_cv = vectorizer_bow_essays.transform(X_cv['preprocessed_essays'])
text_bow_test = vectorizer_bow_essays.transform(X_test['preprocessed_essays'])
print("Shape of matrix after BOW_text_train ",text_bow_train.shape)
print("Shape of matrix after BOW_text_cv ",text_bow_cv.shape)
print("Shape of matrix after BOW_text_test ",text_bow_test.shape)
```

```
Shape of matrix after BOW_text_train  (22445, 8857)
Shape of matrix after BOW_text_cv  (11055, 8857)
Shape of matrix after BOW_text_test  (16500, 8857)
```

Bag of Words(BOW) on `project_title` (Train,Cv,Test)

In [45]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer_bow_titles = CountVectorizer(min_df=10)
vectorizer_bow_titles.fit(X_train['preprocessed_titles'])

title_bow_train = vectorizer_bow_titles.transform(X_train['preprocessed_title
s'])
title_bow_cv = vectorizer_bow_titles.transform(X_cv['preprocessed_titles'])
title_bow_test = vectorizer_bow_titles.transform(X_test['preprocessed_titles'
])
print("Shape of matrix after BOW_title_train ",title_bow_train.shape)
print("Shape of matrix after BOW_title_cv ",title_bow_cv.shape)
print("Shape of matrix after BOW_title_test ",title_bow_test.shape)
```

```
Shape of matrix after BOW_title_train  (22445, 1250)
Shape of matrix after BOW_title_cv  (11055, 1250)
Shape of matrix after BOW_title_test  (16500, 1250)
```

TFIDF Vectorizer on `project_TEXT/ESSAYS` (Train,Cv,Test)

In [46]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essays = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essays.fit(X_train['preprocessed_essays'])

text_tfidf_train = vectorizer_tfidf_essays.transform(X_train['preprocessed_ess
ays'])
text_tfidf_cv = vectorizer_tfidf_essays.transform(X_cv['preprocessed_essays'])
text_tfidf_test = vectorizer_tfidf_essays.transform(X_test['preprocessed_essay
s'])
print("Shape of matrix after tfidf_text_train ",text_tfidf_train.shape)
print("Shape of matrix after tfidf_text_cv ",text_tfidf_cv.shape)
print("Shape of matrix after tfidf_text_test ",text_tfidf_test.shape)
```

```
Shape of matrix after tfidf_text_train  (22445, 8857)
Shape of matrix after tfidf_text_cv  (11055, 8857)
Shape of matrix after tfidf_text_test  (16500, 8857)
```

TFIDF Vectorizer on `project_title` (Train,Cv,Test)

In [47]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(X_train['preprocessed_titles'])

title_tfidf_train = vectorizer_tfidf_title.transform(X_train['preprocessed_tit
les'])
title_tfidf_cv = vectorizer_tfidf_title.transform(X_cv['preprocessed_titles'])
title_tfidf_test = vectorizer_tfidf_title.transform(X_test['preprocessed_title
s'])
print("Shape of matrix after tfidf_title_train ",title_tfidf_train.shape)
print("Shape of matrix after tfidf_title_cv ",title_tfidf_cv.shape)
print("Shape of matrix after tfidf_title_test ",title_tfidf_test.shape)
```

```
Shape of matrix after tfidf_title_train  (22445, 1250)
Shape of matrix after tfidf_title_cv  (11055, 1250)
Shape of matrix after tfidf_title_test  (16500, 1250)
```

In [48]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how
-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [49]:
```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debug
ging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the rea
der
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [1]:
```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('../notebook_env.db')
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarnin
g: paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip insta
ll paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disable
d.  `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarnin
g: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

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 [3]:
```
# 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_st
ate_one_hot_train,teacher_prefix_one_hot_train
               ,project_grade_category_one_hot_train,price_standardized_train,
quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_bow_train,title_bow_train)).tocsr().toarray()
X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_on
e_hot_cv,teacher_prefix_one_hot_cv
               ,project_grade_category_one_hot_cv,price_standardized_cv,quanti
ty_standardized_cv
               ,teacher_number_of_previously_posted_projects_standardized_cv,t
ext_bow_cv,title_bow_cv)).tocsr().toarray()
X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_stat
e_one_hot_test,teacher_prefix_one_hot_test
               ,project_grade_category_one_hot_test,price_standardized_test,qu
antity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_bow_test,title_bow_test)).tocsr().toarray()

print("Final Data matrix on BOW")
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 on BOW
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
================================================================================
=======================
```

In [4]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_tr = scaler.fit_transform(X_tr,y_train)
X_cr = scaler.transform(X_cr)
X_te = scaler.transform(X_te)
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
```

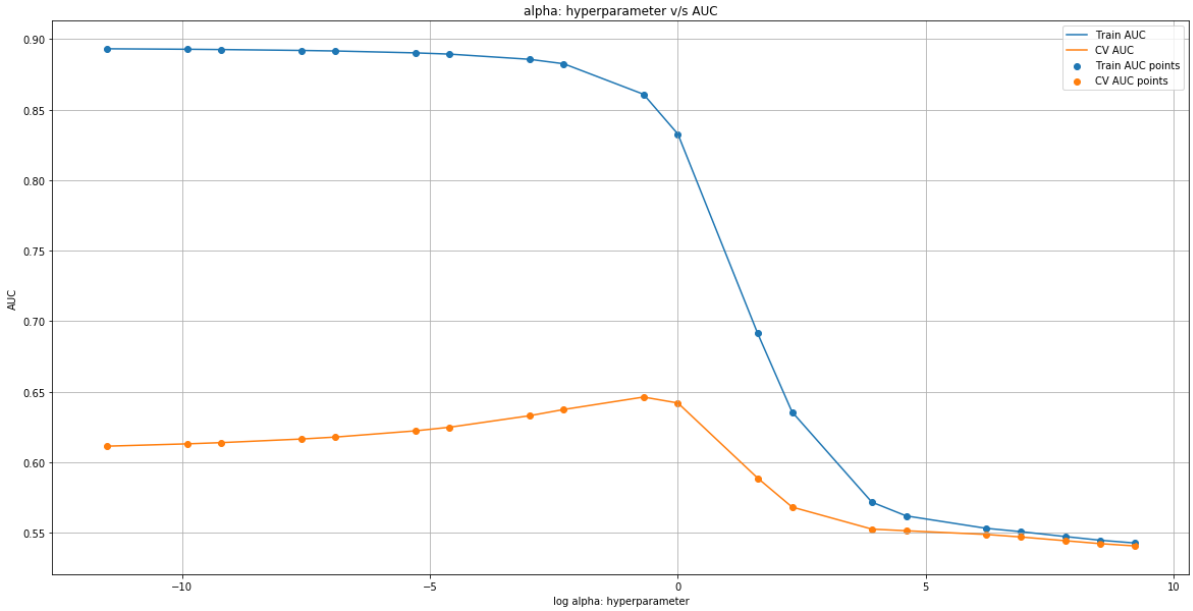
```
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
```

1.1 Method 1: Simple for loop (if you are having memory limitations use this)

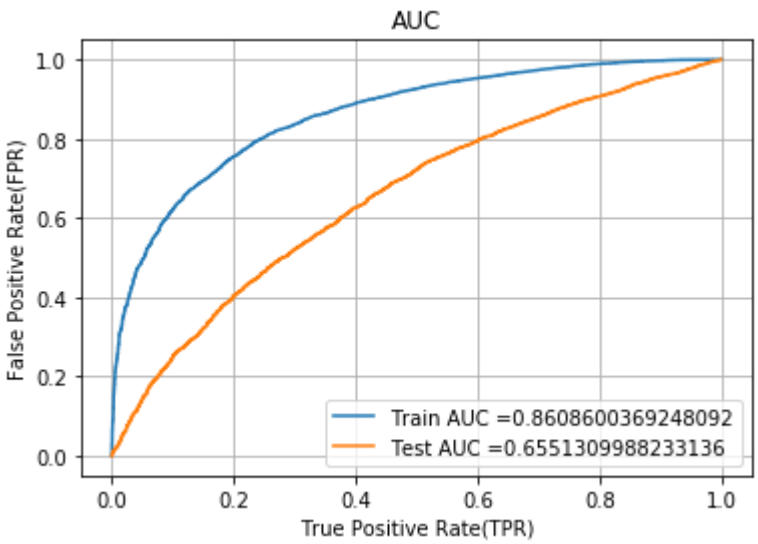
In [5]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs

    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 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [6]:
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math
from sklearn.model_selection import RandomizedSearchCV

train_auc = []
cv_auc = []
log_alphas = []
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5
, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior=[0.5,0.5])
    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 e
stimates of the positive 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)

log_alphas = np.array(log_alphas)
alphas = np.array(alphas)

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.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()

# print(len(log_alphas))
# print(log_alphas.shape)
# print(train_auc.shape)
```

```
100%|
| 20/20 [00:33<00:00,  1.51s/it]
100%|
| 20/20 [00:00<?, ?it/s]
```



In [7]:
```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.h
tml#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

nb_bow = MultinomialNB(alpha = 0.5,class_prior=[0.5,0.5])
nb_bow.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

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

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

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



In [8]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

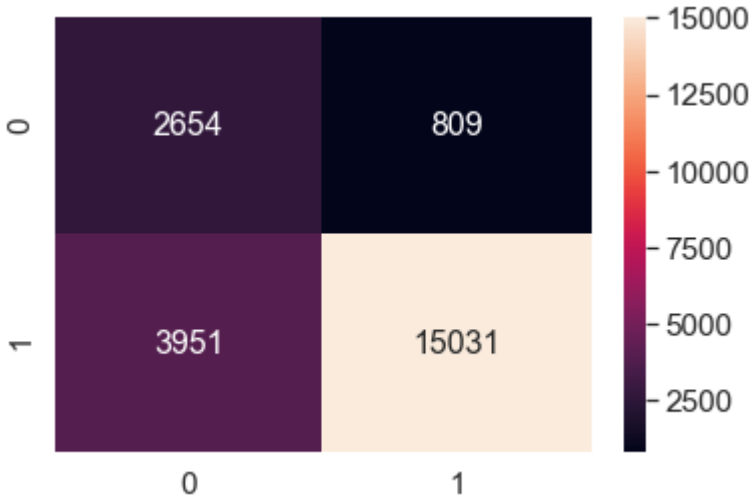
In [9]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
==============================================================================
========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.6068681325618965 for threshold 0.49
[[ 2654   809]
 [ 3951 15031]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3774556192431624 for threshold 0.422
[[ 1045  1501]
 [ 2968 10986]]
```

In [10]:
```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_trai
n_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.6068681325618965 for threshold 0.49
```

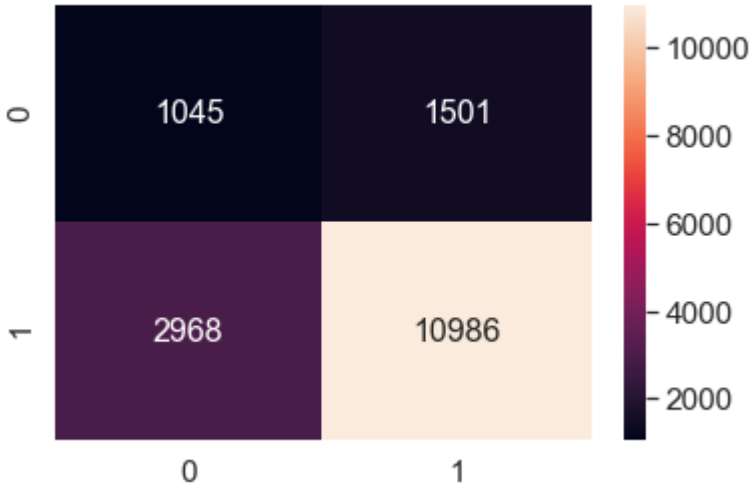
Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f127fb4b38>
```



In [11]:
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_p
red, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.3774556192431624 for threshold 0.422
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1f128140748>
```



In [11]:
```
# Please write all the code with proper documentation
```

2.4.1.1 Top 10 important features of positive class from SET 1

In [2]:
```
# 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_st
ate_one_hot_train,teacher_prefix_one_hot_train
               ,project_grade_category_one_hot_train,price_standardized_train,
quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_bow_train,title_bow_train)).tocsr().toarray()
X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_on
e_hot_cv,teacher_prefix_one_hot_cv
               ,project_grade_category_one_hot_cv,price_standardized_cv,quanti
ty_standardized_cv
               ,teacher_number_of_previously_posted_projects_standardized_cv,t
ext_bow_cv,title_bow_cv)).tocsr().toarray()
X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_stat
e_one_hot_test,teacher_prefix_one_hot_test
               ,project_grade_category_one_hot_test,price_standardized_test,qu
antity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_bow_test,title_bow_test)).tocsr().toarray()

print("Final Data matrix on BOW")
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 on BOW
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
==============================================================================
=======================
```

In [3]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_tr = scaler.fit_transform(X_tr,y_train)
X_cr = scaler.transform(X_cr)
X_te = scaler.transform(X_te)
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
```

```
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
```

In [4]:
```
from sklearn.naive_bayes import MultinomialNB


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

Out[4]:
```
MultinomialNB(alpha=0.5, class_prior=[0.5, 0.5], fit_prior=True)
```

In [5]:
```
# bow_features_probs1 = {}
# bow_features_probs0 = {}
# for a in range(10212) :
#     bow_features_probs1[a] = nb_bow.feature_log_prob_[1,a]
# bow_features_probs = {}
# for b in range(10212) :
#     bow_features_probs0[b] = nb_bow.feature_log_prob_[0,a]
# # print((bow_features_probs1.items[:3]))
# c1=0
# for k,v in bow_features_probs1.items():
#     print(k, v)
#     c1 = c1 + 1
#     if(c1==5):
#         break

# print("="*100)
# # print((bow_features_probs0))
# c2=0
# for k,v in bow_features_probs0.items():
#     print(k, v)
#     c2 = c2 + 1
#     if(c2==5):
#         break
```

In [6]:
```
bow_features_probs1 = []
for a in range(10209) :
    b = nb_bow.feature_log_prob_[1,a]
    bow_features_probs1.append(b)

len(bow_features_probs1)
```

Out[6]: 10209

In [7]:
```
bow_features_names = []
```

In [8]:
```
for a in vectorizer_cat.get_feature_names() :
    bow_features_names.append(a)
```

In [9]:
```
for a in vectorizer_sub_cat.get_feature_names() :
    bow_features_names.append(a)
```

In [10]:
```
for a in vectorizer_state.get_feature_names() :
    bow_features_names.append(a)
```

In [11]:
```
for a in vectorizer_teacherprefix.get_feature_names() :
    bow_features_names.append(a)
```

In [12]:
```
for a in vectorizer_projectgrade.get_feature_names() :
    bow_features_names.append(a)
```

In [13]:
```
bow_features_names.append("price")
bow_features_names.append("quantity")
bow_features_names.append("teacher_number_of_previously_posted")
```

In [14]:
```
for a in vectorizer_bow_essays.get_feature_names() :
    bow_features_names.append(a)
```

In [15]:
```
for a in vectorizer_bow_titles.get_feature_names() :
    bow_features_names.append(a)
len(bow_features_names)
```

Out[15]: 10209

In [16]:
```
final_features_bow_df_pos = pd.DataFrame({'feature_prob_estimates' : bow_featu
res_probs1,'feature_names' : bow_features_names})
```

In [17]:
```
final_features_bow_df_pos.sort_values(by = ['feature_prob_estimates'], ascendi
ng = False,inplace=True)
```

In [18]:
```
final_features_bow_df_pos.head(10)
```

Out[18]:

	feature_prob_estimates	feature_names
92	-4.215421	Mrs
8	-4.287486	Literacy_Language
98	-4.474116	GradesPreK-2
7	-4.574186	Math_Science
93	-4.601107	Ms
95	-4.653046	Grades3-5
38	-4.705384	Literacy
7766	-4.880016	students
37	-4.951733	Mathematics
36	-5.158836	Literature_Writing

In [19]:
```
# Please write all the code with proper documentation
```

2.4.1.2 Top 10 important features of negative class from SET 1

In [20]:
```
bow_features_probs2 = []
for a in range(10209) :
    bb = nb_bow.feature_log_prob_[0,a]
    bow_features_probs2.append(bb)

# (bow_features_probs)
```

In [21]:
```
final_features_bow_df_neg = pd.DataFrame({'feature_prob_estimates' : bow_featu
res_probs2,'feature_names' : bow_features_names})
```

In [22]:
```
final_features_bow_df_neg.sort_values(by = ['feature_prob_estimates'], ascendi
ng = False,inplace=True)
```

In [23]:
```
final_features_bow_df_neg.head(10)
```

Out[23]:

	feature_prob_estimates	feature_names
92	-4.253758	Mrs
8	-4.402431	Literacy_Language
98	-4.458514	GradesPreK-2
7	-4.493592	Math_Science
93	-4.572391	Ms
95	-4.707219	Grades3-5
38	-4.874765	Literacy
37	-4.927322	Mathematics
7766	-4.941158	students
36	-5.226500	Literature_Writing

In [0]:
```
# Please write all the code with proper documentation
```

In [2]:
```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('../notebook_env.db')
```

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [2]:
```
# 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_st
ate_one_hot_train,teacher_prefix_one_hot_train
               ,project_grade_category_one_hot_train,price_standardized_train,
quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_tfidf_train,title_tfidf_train)).tocsr().toarray()
X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_on
e_hot_cv,teacher_prefix_one_hot_cv
               ,project_grade_category_one_hot_cv,price_standardized_cv,quanti
ty_standardized_cv
               ,teacher_number_of_previously_posted_projects_standardized_cv,t
ext_tfidf_cv,title_tfidf_cv)).tocsr().toarray()
X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_stat
e_one_hot_test,teacher_prefix_one_hot_test
               ,project_grade_category_one_hot_test,price_standardized_test,qu
antity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_tfidf_test,title_tfidf_test)).tocsr().toarray()

print("Final Data matrix on TFIDF")
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 on TFIDF
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
==============================================================================
=======================
```

In [3]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_tr = scaler.fit_transform(X_tr,y_train)
X_cr = scaler.transform(X_cr)
X_te = scaler.transform(X_te)
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
```

```
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
```

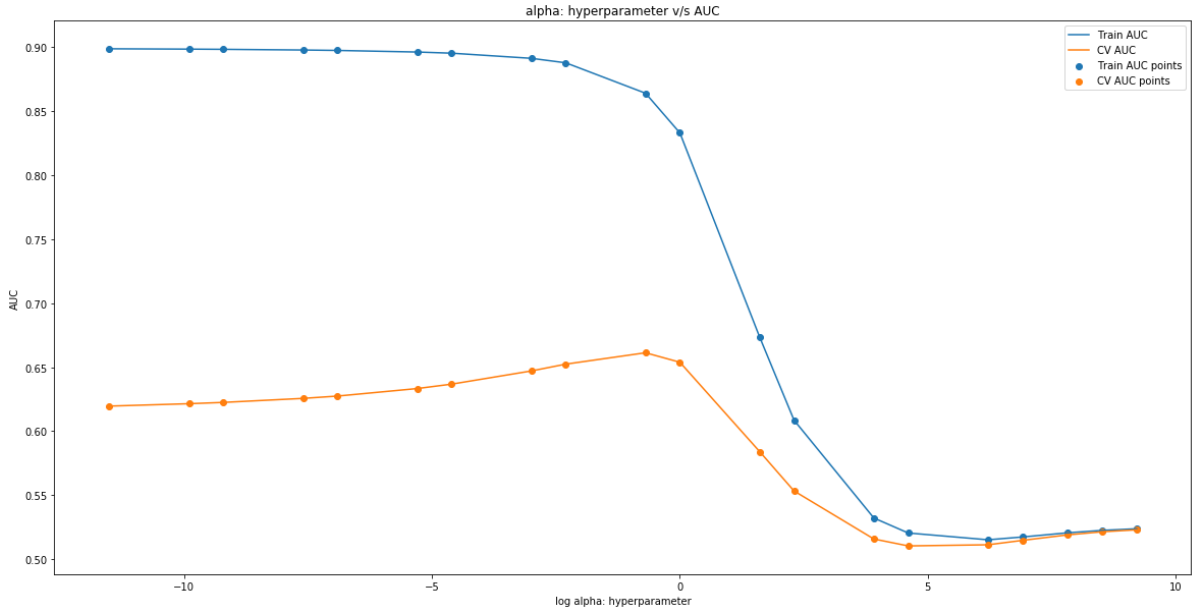
In [ ]:

1.1 Method 1: Simple for loop (if you are having memory limitations use this)

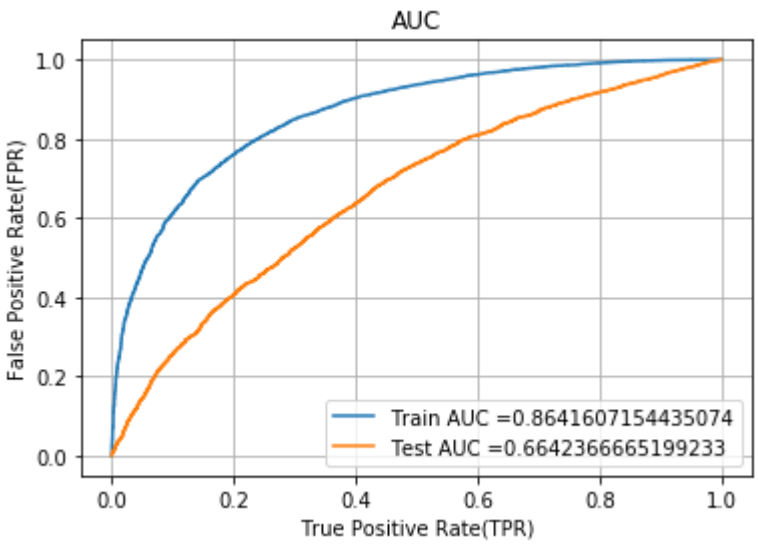
In [4]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs

    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 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [5]:
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math
from sklearn.model_selection import RandomizedSearchCV

train_auc = []
cv_auc = []
log_alphas = []
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5
, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior=[0.5,0.5])
    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 e
stimates of the positive 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)

log_alphas = np.array(log_alphas)
alphas = np.array(alphas)

plt.figure(figsize=(20,10))
plt.grid()
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.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()

# print(len(log_alphas))
# print(log_alphas.shape)
# print(train_auc.shape)
```

```
100%|
| 20/20 [00:31<00:00,  1.52s/it]
100%|
| 20/20 [00:00<?, ?it/s]
```



In [6]:
```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.h
tml#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

nb_tfidf = MultinomialNB(alpha = 0.5,class_prior=[0.5,0.5])
nb_tfidf.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(nb_tfidf, X_tr)
y_test_pred = batch_predict(nb_tfidf, 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_tp
r)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [7]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

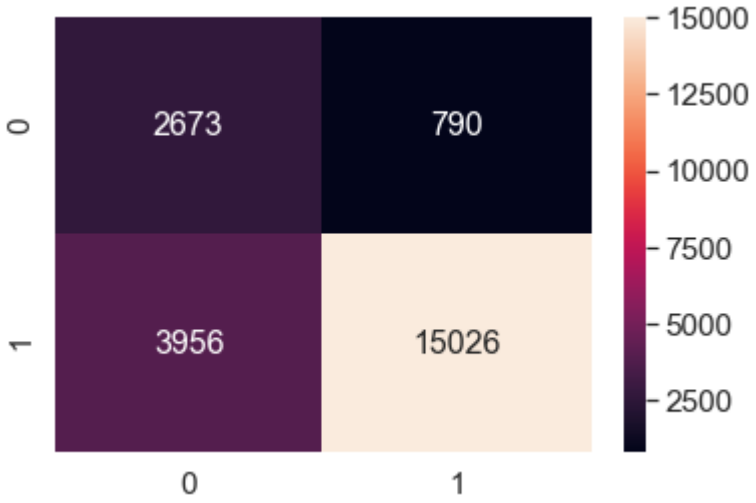
In [8]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
==============================================================================
========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.6110093873451795 for threshold 0.485
[[ 2673   790]
 [ 3956 15026]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.38422744871179804 for threshold 0.376
[[  984  1562]
 [ 2575 11379]]
```

In [9]:
```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_trai
n_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.6110093873451795 for threshold 0.485
```

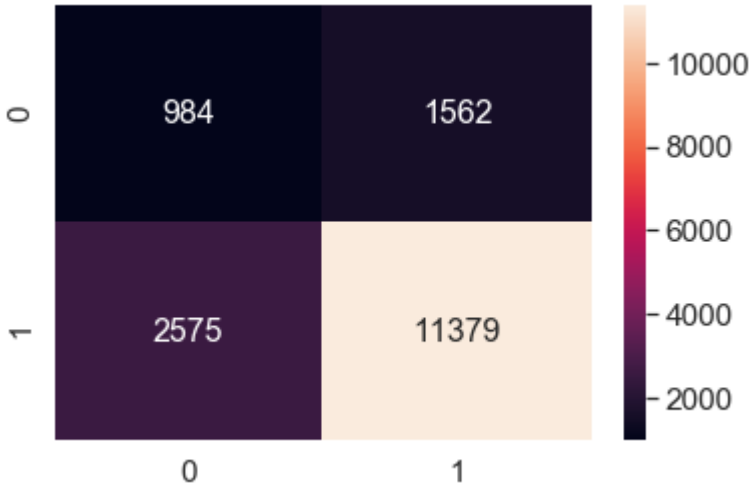
Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25de9172be0>
```



In [10]:
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_p
red, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.38422744871179804 for threshold 0.376
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x25de92fe320>
```



In [11]:
```
# Please write all the code with proper documentation
```

2.4.2.1 Top 10 important features of positive class from SET 2

In [3]:
```
# 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_st
ate_one_hot_train,teacher_prefix_one_hot_train
               ,project_grade_category_one_hot_train,price_standardized_train,
quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_tfidf_train,title_tfidf_train)).tocsr().toarray()
X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_on
e_hot_cv,teacher_prefix_one_hot_cv
               ,project_grade_category_one_hot_cv,price_standardized_cv,quanti
ty_standardized_cv
               ,teacher_number_of_previously_posted_projects_standardized_cv,t
ext_tfidf_cv,title_tfidf_cv)).tocsr().toarray()
X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_stat
e_one_hot_test,teacher_prefix_one_hot_test
               ,project_grade_category_one_hot_test,price_standardized_test,qu
antity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_tfidf_test,title_tfidf_test)).tocsr().toarray()

print("Final Data matrix on TFIDF")
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 on TFIDF
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
==============================================================================
=======================
```

In [4]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_tr = scaler.fit_transform(X_tr,y_train)
X_cr = scaler.transform(X_cr)
X_te = scaler.transform(X_te)
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
```

```
(22445, 10209) (22445,)
(11055, 10209) (11055,)
(16500, 10209) (16500,)
```

In [5]:
```
from sklearn.naive_bayes import MultinomialNB


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

Out[5]:
```
MultinomialNB(alpha=0.5, class_prior=[0.5, 0.5], fit_prior=True)
```

In [6]:
```
tfidf_features_probs1 = []
for a in range(10209) :
    b = nb_tfidf.feature_log_prob_[1,a]
    tfidf_features_probs1.append(b)

len(tfidf_features_probs1)
```

Out[6]: 10209

In [7]:
```
tfidf_features_names = []
```

In [8]:
```
for a in vectorizer_cat.get_feature_names() :
    tfidf_features_names.append(a)
```

In [9]:
```
for a in vectorizer_sub_cat.get_feature_names() :
    tfidf_features_names.append(a)
```

In [10]:
```
for a in vectorizer_state.get_feature_names() :
    tfidf_features_names.append(a)
```

In [11]:
```
for a in vectorizer_teacherprefix.get_feature_names() :
    tfidf_features_names.append(a)
```

In [12]:
```
for a in vectorizer_projectgrade.get_feature_names() :
    tfidf_features_names.append(a)
```

In [13]:
```
tfidf_features_names.append("price")
tfidf_features_names.append("quantity")
tfidf_features_names.append("teacher_number_of_previously_posted")
```

In [14]:
```
for a in vectorizer_tfidf_essays.get_feature_names() :
    tfidf_features_names.append(a)
```

In [15]:
```
for a in vectorizer_tfidf_title.get_feature_names() :
    tfidf_features_names.append(a)
len(tfidf_features_names)
```

Out[15]: 10209

In [16]:
```
final_features_tfidf_df_pos = pd.DataFrame({'feature_prob_estimates' : tfidf_f
eatures_probs1,'feature_names' : tfidf_features_names})
```

In [17]:
```
final_features_tfidf_df_pos.sort_values(by = ['feature_prob_estimates'], ascen
ding = False,inplace=True)
```

In [18]:
```
final_features_tfidf_df_pos.head(10)
```

Out[18]:

	feature_prob_estimates	feature_names
92	-4.220127	Mrs
8	-4.292192	Literacy_Language
98	-4.478822	GradesPreK-2
7766	-4.573659	students
7	-4.578892	Math_Science
93	-4.605813	Ms
95	-4.657751	Grades3-5
38	-4.710090	Literacy
37	-4.956439	Mathematics
36	-5.163542	Literature_Writing

In [19]:
```
# Please write all the code with proper documentation
```

2.4.2.2 Top 10 important features of negative class from SET 2

In [20]:
```
tfidf_features_probs2 = []
for a in range(10209) :
    bb = nb_tfidf.feature_log_prob_[0,a]
    tfidf_features_probs2.append(bb)

# (bow_features_probs)
```

In [21]:
```
final_features_tfidf_df_neg = pd.DataFrame({'feature_prob_estimates' : tfidf_f
eatures_probs2,'feature_names' : tfidf_features_names})
```

In [22]:
```
final_features_tfidf_df_neg.sort_values(by = ['feature_prob_estimates'], ascen
ding = False,inplace=True)
```

In [23]:
```
final_features_tfidf_df_neg.head(10)
```

Out[23]:

	feature_prob_estimates	feature_names
92	-4.294053	Mrs
8	-4.442726	Literacy_Language
98	-4.498809	GradesPreK-2
7	-4.533887	Math_Science
93	-4.612686	Ms
7766	-4.616482	students
95	-4.747514	Grades3-5
38	-4.915060	Literacy
37	-4.967617	Mathematics
36	-5.266795	Literature_Writing

In [0]:
```
# Please write all the code with proper documentation
```

3. Conclusions

In [0]:
```
# Please compare all your models using Prettytable library
```

In [24]:
```
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 inst
all prettytable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter:Alpha", "AUC"]
x.add_row(["BOW", "Naive Bayes", 0.5, 0.65])
x.add_row(["TFIDF", "Naive Bayes", 0.5, 0.66])
print(x)
```

```
+------------+-------------+-----------------------+------+
| Vectorizer |    Model    | Hyper Parameter:Alpha | AUC  |
+------------+-------------+-----------------------+------+
|    BOW     | Naive Bayes |          0.5          | 0.65 |
|   TFIDF    | Naive Bayes |          0.5          | 0.66 |
+------------+-------------+-----------------------+------+
```

In [ ]: