

DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
<code>project_id</code>	A unique identifier for the proposed project. Example: p036502
<code>project_title</code>	Title of the project. Examples: <ul style="list-style-type: none">• Art Will Make You Happy!• First Grade Fun
<code>project_grade_category</code>	Grade level of students for which the project is targeted. One of the following enumerated values: <ul style="list-style-type: none">• Grades PreK-2• Grades 3-5• Grades 6-8• Grades 9-12
<code>project_subject_categories</code>	One or more (comma-separated) subject categories for the project from the following enumerated list of values: <ul style="list-style-type: none">• Applied Learning• Care & Hunger• Health & Sports• History & Civics• Literacy & Language• Math & Science• Music & The Arts• Special Needs• Warmth Examples: <ul style="list-style-type: none">• Music & The Arts• Literacy & Language, Math & Science
<code>school_state</code>	State where school is located (Two-letter U.S. postal code). Example: WY
<code>project_subject_subcategories</code>	One or more (comma-separated) subject subcategories for the project. Examples: <ul style="list-style-type: none">• Literacy

Feature	Description
<code>project_resource_summary</code>	An explanation of the resources needed for the project. Example: <ul style="list-style-type: none"> My students need hands on literacy materials to manage sensory needs!
<code>project_essay_1</code>	First application essay*
<code>project_essay_2</code>	Second application essay*
<code>project_essay_3</code>	Third application essay*
<code>project_essay_4</code>	Fourth application essay*
<code>project_submitted_datetime</code>	Datetime when project application was submitted. Example: 2016-04-28 12:43:56.245
<code>teacher_id</code>	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4ff1c15c56
<code>teacher_prefix</code>	Teacher's title. One of the following enumerated values: <ul style="list-style-type: none"> nan Dr. Mr. Mrs. Ms. Teacher.
<code>teacher_number_of_previously_posted_projects</code>	Number of project applications previously submitted by the same teacher. Example: 2

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

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

Feature	Description
<code>id</code>	A <code>project_id</code> value from the <code>train.csv</code> file. Example: p036502
<code>description</code>	Description of the resource. Example: Tenor Saxophone Reeds, Box of 25
<code>quantity</code>	Quantity of the resource required. Example: 3
<code>price</code>	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
<code>project_is_approved</code>	A binary flag indicating whether DonorsChoose approved the project. A value of 0 indicates the project was not approved, and a value of 1 indicates the project was approved.

Notes on the Essay Data

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

- `__project_essay_1__` "Introduce us to your classroom"
- `__project_essay_2__` "Tell us more about your students"
- `__project_essay_3__` "Describe how your students will use the materials you're requesting"
- `__project_essay_3__` "Close by sharing why your project will make a difference"

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

- `__project_essay_1__` "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful"

your neighborhood, and your school are all helpful.

- __project_essay_2__: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [1]:

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

os.chdir('C:/Users/kingsubham27091995/Desktop/AppliedAiCouse/DonorsChoose')
```

1.1 Reading Data

In [2]:

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

In [3]:

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

Number of data points in train data (109248, 17)

```
-----
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher number of previously posted projects' 'project is approved']
```

In [4]:

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

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

Out[4]:

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

In [5]:

```
project_grade_category = []

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

In [6]:

```
project grade category[0:5]
```

Out[6]:

```
['Grades PreK-2', 'Grades 6-8', 'Grades 6-8', 'Grades PreK-2', 'Grades PreK-2']
```

In [7]:

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

In [8]:

```
project_data["project grade category"] = project_grade_category
```

In [9]:

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

Out [9] :

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	proj
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Lite
1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Hist Spec
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Hea

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Lite Sci
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Mat

1.2 preprocessing of project_subject_categories

In [10]:

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

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

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

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

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

1.3 preprocessing of project_subject_subcategories

In [11]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
sub_cat_list = []
for i in sub_categories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the category 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')
```

```

.e removing 'The')
    j = j.replace(' ', '') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
    temp +=j.strip()+" #" abc ".strip() will return "abc", remove the trailing spaces
    temp = temp.replace('&', '_')
    sub_cat_list.append(temp.strip())

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

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

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

```

Text Preprocessing

Finding number of words in title and introducing it in a new column

- This can be used as Numerical Feature for Vectorisation

In [12]:

```

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

```

In [13]:

```

project_data["title_word_count"] = title_word_count
project_data.head(5)

```

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Edu Sup Eng Lea Hon
1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Wai Proj Hur Lea
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Soc Equ AW Mid Stu.
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Tec Kinc
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Inte Too

Combining 4 Project essays into 1 Essay feature

In [14]:

```
# 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)
```

Finding number of words in essay and introducing it in a new column

- This can be used as Numerical Feature for Vectorisation

In [15]:

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

In [16]:

```
project_data["essay_word_count"] = essay_word_count
project_data.head(5)
```

Out[16]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Edu Sup Eng Lea Hor
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Wai Proj Hur Lea
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Soc Equ AW Mid Stu.
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Tec Kinc
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Inte Too

Splitting Project_Data into Train and Test Datasets

In [17]:

```
from sklearn.model_selection import train_test_split

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

We don't need the 'project_is_approved' feature now

In [18]:

```
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)
```

Printing some random essays

In [19]:

```
# 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 alongside 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 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\n\r\nannan

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 a 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.

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 themed room for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free 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 them 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 [20]:

```
# 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 [21]:

```

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

```

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 them 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 [22]:

```

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

```

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. The materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come 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 them 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 old deserves. nannan

In [23]:

```

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

```

My kindergarten students have varied disabilities ranging from speech and language delays cognitive delays gross fine motor delays to autism They are eager beavers and always strive to work their hardest working past their limitations The materials we have are the ones I seek out for my 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 them 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 old deserves nannan

In [24]:

```

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

```

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

Preprocessed Training data (Essay Feature)

In [25]:

```
# Combining all the above

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

100%|██| 49041/49041 [00:37<00:00, 1318.97it/s]

In [26]:

```
# after preprocessing
preprocessed_essays_train[1000]
```

Out[26]:

'school largest range highest lowest socio economic families school district across student popula
tion 20 languages spoken home although students access books personal libraries many not additiona
lly many students cannot even access public library due family work commitments books available st
udents school help teach powerful unit encourage students make connections incredible figures like
malala yousafzai school implements common core standards committed reading nonfiction
informational materials imperative students access individual copy malala copy students able read
independently make personal connections prepare class discussions writing assignments read malala
students introduced world problems contemplate possible solutions engage critical thinking read di
scuss write malala story world problems unit one school powerful 8th grade units cross curricular
unit partnering social studies concludes presentation project revolving around world problems solu
tions nannan'

Preprocessed Test data (Essay Feature)

In [27]:

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

100%|██| 36052/36052 [00:27<00:00, 1307.07it/s]

In [28]:

```
# after preprocessing
preprocessed_essays_test[1000]
```

Out[28]:

'love learning hunger know important things hope inspire students students grasp complex content first capture interest ideas supersede language questions like classify categorize world important plan embark project acting things help us solve problems find answers questions without adult intervention access many different types exploration art materials help student organize express thinking learning high complex levels students remarkably engaged questions like especially chance document learning multiple ways students move beyond rote retelling stories recreating creating tales able reference back stories read award winning authors become author stories work plan produce edit stories oral written students perform stories peers partner class write invitations schedules events posters advertising performances tickets entry programs students take away also create versions stories bound classroom library color ink makes color reproduction stories possible include partner reading center nannan'

Preprocessed Cross Validation data (Essay Feature)

In [29]:

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

100%|██| 24155/24155 [00:18<00:00, 1282.86it/s]

In [30]:

```
# after preprocessing
preprocessed_essays_cv[1000]
```

Out[30]:

'students active love working one another able puzzles way organize puzzle help classroom thrive hands activities expand learning puzzle time throughout day creates great brain break learning experience young age students hands active day come ready school day excitement going day love learning enjoy school students fun working others create new things kindergarten teacher students active engaged different learning techniques student varies way make classroom wonderful place week set aside special time students complete puzzles not enjoy working others also improving cognitive physical social skills without even knowing puzzle time allows children grow students children society love giving opportunity succeed different activities materials help addition educational puzzles assist students learning numbers letters nannan'

Preprocessing Training data(Titles Feature)

In [31]:

```
preprocessed_titles_train = []

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

100%|██| 49041/49041 [00:02<00:00, 23293.76it/s]

In [32]:

```
preprocessed_titles_train[1000]
```

Out[32]:

```
'learning about world problems through i am malala'
```

Preprocessing Test data(Titles Feature)

In [33]:

```
preprocessed_titles_test = []

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

100%|██| 36052/36052 [00:01<00:00, 28213.15it/s]

In [34]:

```
preprocessed_titles_test[1000]
```

Out[34]:

```
'tell me tale blocks'
```

Preprocessing Cross Validation data(Titles Feature)

In [35]:

```
preprocessed_titles_cv = []

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

100%|██| 24155/24155 [00:00<00:00, 25717.88it/s]

In [36]:

```
preprocessed_titles_cv[1000]
```

Out[36]:

```
'puzzle time'
```

1.10 Preparing data for models

In [37]:

```
project_data.columns
```

Out[37]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',  
      'project_submitted_datetime', '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',  
      'project_grade_category', 'clean_categories', 'clean_subcategories',  
      'title_word_count', 'essay', 'essay_word_count'],  
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)
- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
- title_word_count : numerical
- essay_word_count : numerical

1.11 Vectorizing Categorical data

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

One Hot Encoding - (Clean Categories Feature)

In [38]:

```
# we use count vectorizer to convert the values into one  
  
from sklearn.feature_extraction.text import CountVectorizer  
  
vectorizer_proj = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)  
vectorizer_proj.fit(X_train['clean_categories'].values)  
  
categories_one_hot_train = vectorizer_proj.transform(X_train['clean_categories'].values)  
categories_one_hot_test = vectorizer_proj.transform(X_test['clean_categories'].values)  
categories_one_hot_cv = vectorizer_proj.transform(X_cv['clean_categories'].values)  
  
print(vectorizer_proj.get_feature_names())
```

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

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

One Hot Encoding - (Clean Sub-Categories Feature)

In [39]:

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

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

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

print(vectorizer_sub_proj.get_feature_names())

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

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

One Hot Encoding - (School States Feature)

In [40]:

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

In [41]:

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

In [42]:

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

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

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

```

print(vectorizer_states.get_feature_names())

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

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

```

One Hot Encoding - (Project Grade Category Feature)

In [43]:

```

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

```

In [44]:

```

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

```

In [45]:

```

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

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

project_grade_categories_one_hot_train =
vectorizer_grade.transform(X_train['project_grade_category'].values)
project_grade_categories_one_hot_test = vectorizer_grade.transform(X_test['project_grade_category'
].values)
project_grade_categories_one_hot_cv = vectorizer_grade.transform(X_cv['project_grade_category'].va
lues)

print(vectorizer_grade.get_feature_names())

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

['Grades_9-12', 'Grades_6-8', 'Grades_3-5', 'Grades_PreK-2']
Shape of matrix of Train data after one hot encoding (49041, 4)
Shape of matrix of Test data after one hot encoding (36052, 4)
Shape of matrix of Cross Validation data after one hot encoding (24155, 4)

```

One Hot Encoding - (Teacher Prefix Feature)

In [46]:

```

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

```


In [47]:

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

In [48]:

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

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

teacher_prefix_categories_one_hot_train = vectorizer_teacher.transform(X_train['teacher_prefix'].v
alues.astype("U"))
teacher_prefix_categories_one_hot_test =
vectorizer_teacher.transform(X_test['teacher_prefix'].values.astype("U"))
teacher_prefix_categories_one_hot_cv = vectorizer_teacher.transform(X_cv['teacher_prefix'].values.
astype("U"))

print(vectorizer_teacher.get_feature_names())

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

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

1.12 Vectorizing Text data

Bag of words - Training Data (Essays Feature)

In [49]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(preprocessed_essays_train)
text_bow_train = vectorizer_bow_essay.transform(preprocessed_essays_train)
print("Shape of matrix after one hot encoding ",text_bow_train.shape)
```

Shape of matrix after one hot encoding (49041, 12042)

Bag of words - Test Data (Essays Feature)

In [50]:

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

Shape of matrix after one hot encoding (36052, 12042)

Bag of words - Cross Validation Data (Essays Feature)

In [51]:

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

Shape of matrix after one hot encoding (24155, 12042)

Bag of words - Train Data - (Titles Feature)

In [52]:

```
vectorizer_bow_title = CountVectorizer(min_df=10)

vectorizer_bow_title.fit(preprocessed_titles_train)

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

Shape of matrix after one hot encoding (49041, 2119)

Bag of words - Test Data - (Titles Feature)

In [53]:

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

Shape of matrix after one hot encoding (36052, 2119)

Bag of words - Cross Validation Data -(Titles Feature)

In [54]:

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

Shape of matrix after one hot encoding (24155, 2119)

TFIDF - Training Data - (Essays Feature)

In [55]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(preprocessed_essays_train)
text_tfidf_train = vectorizer_tfidf_essay.transform(preprocessed_essays_train)
print("Shape of matrix after one hot encoding ",text_tfidf_train.shape)
```

Shape of matrix after one hot encoding (49041, 12042)

TFIDF - Test Data -(Essays Feature)

In [56]:

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

Shape of matrix after one hot encoding (36052, 12042)

TFIDF - Cross Validation Data - (Essays Feature)

In [57]:

In [57]:

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

Shape of matrix after one hot encoding (24155, 12042)

TFIDF - Train Data - (Titles Feature)

In [58]:

```
vectorizer_tfidf_titles = TfidfVectorizer(min_df=10)

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

Shape of matrix after one hot encoding (49041, 2119)

TFIDF - Test Data - (Titles Feature)

In [59]:

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

Shape of matrix after one hot encoding (36052, 2119)

TFIDF - Cross Validation Data - (Titles Feature)

In [60]:

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

Shape of matrix after one hot encoding (24155, 2119)

Vectorizing Numerical features

Normalise Numerical values instead of Standardising else it give Error that 'Negative Values not allowed'

- Normalising it will bound the data to 0 and 1

Price Feature

In [61]:

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

Out[61]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [62]:

```
#Now join Price data to Train,Test and Cross Validation Data
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
```

In [63]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

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

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

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

Quantity Feature

In [64]:

```
normalizer = Normalizer()

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

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

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

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

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

Number of Projects previously proposed by Teacher Feature

In [65]:

```
normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
```

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

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

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

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

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

Title word Count Feature

In [66]:

```
normalizer = Normalizer()

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

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

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

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

Essay word Count

In [67]:

```
normalizer_ess_count = Normalizer()

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

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

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

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

Assignment 4: Naive Bayes

1. Apply Multinomial NaiveBayes on these feature sets

- **Set 1:** categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
- **Set 2:** categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

2. The hyper paramter tuning(find best Alpha)

- Find the best hyper parameter which will give the maximum [AUC](#) value
- Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
- Find the best hyper paramter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. Feature importance

- Find the top 10 features of positive class and top 10 features of negative class for both feature sets **Set 1** and **Set 2** using values of `feature_log_prob_` parameter of [MultinomialNB](#) and print their corresponding feature names

4. Representation of results

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
- Along with plotting ROC curve, you need to print the [confusion matrix](#) with predicted and original labels of test data points. Please visualize your confusion matrices using [seaborn heatmaps](#).

5. Conclusion

- [You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link](#)

2. Naive Bayes

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

In [247]:

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

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

In [248]:

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

```
Final Data matrix
(49041, 14266) (49041,)
(24155, 14266) (24155,)
(36052, 14266) (36052,)
```

In [249]:

In [250]:

```
# ===== Multinomial Naive Bayes with alpha = optimal_alpha
# =====
# instantiate learning model alpha = optimal_alpha
bn_optimal = MultinomialNB(alpha = optimal_alpha)

# fitting the model
bn_optimal.fit(X_tr, y_train)
```

```
# predict the response
predictions = bn_optimal.predict(X_te)

# evaluate accuracy
acc = accuracy_score(y_test, predictions) * 100
print('\n\nThe Test Accuracy of the Bernoulli naive Bayes classifier for alpha = %.3f is %f%%' % (optimal_alpha, acc))

# Variables that will be used for making table in Conclusion part of this assignment
bow_multinomial_alpha = optimal_alpha
bow_multinomial_train_acc = max(cv_scores)*100
bow_multinomial_test_acc = acc
```

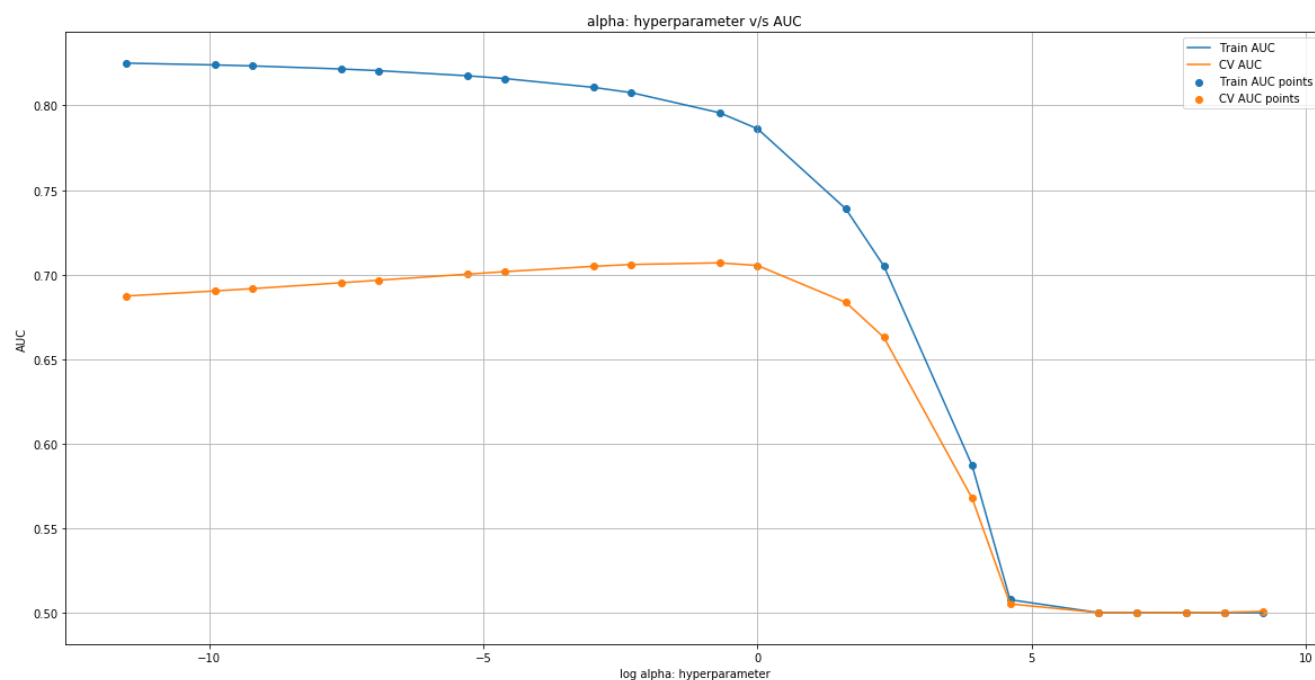
The Test Accuracy of the Bernoulli naive Bayes classifier for alpha = 1.000 is 74.253856%

In [222]:

```
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()
```



Summary :

1. Values ranging between 10^{-4} to 10^4 for alpha parameter was considered.
2. Log of Alphas was plotted on the X axis with the CV Error or AUC values on the Y axis.

B) GridsearchCV using (10 fold cross validation)

In [83]:

```
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB()
```



```
parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]}

clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc')

clf.fit(X_tr, y_train)

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

In [84]:

```
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]
log_alphas = []

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

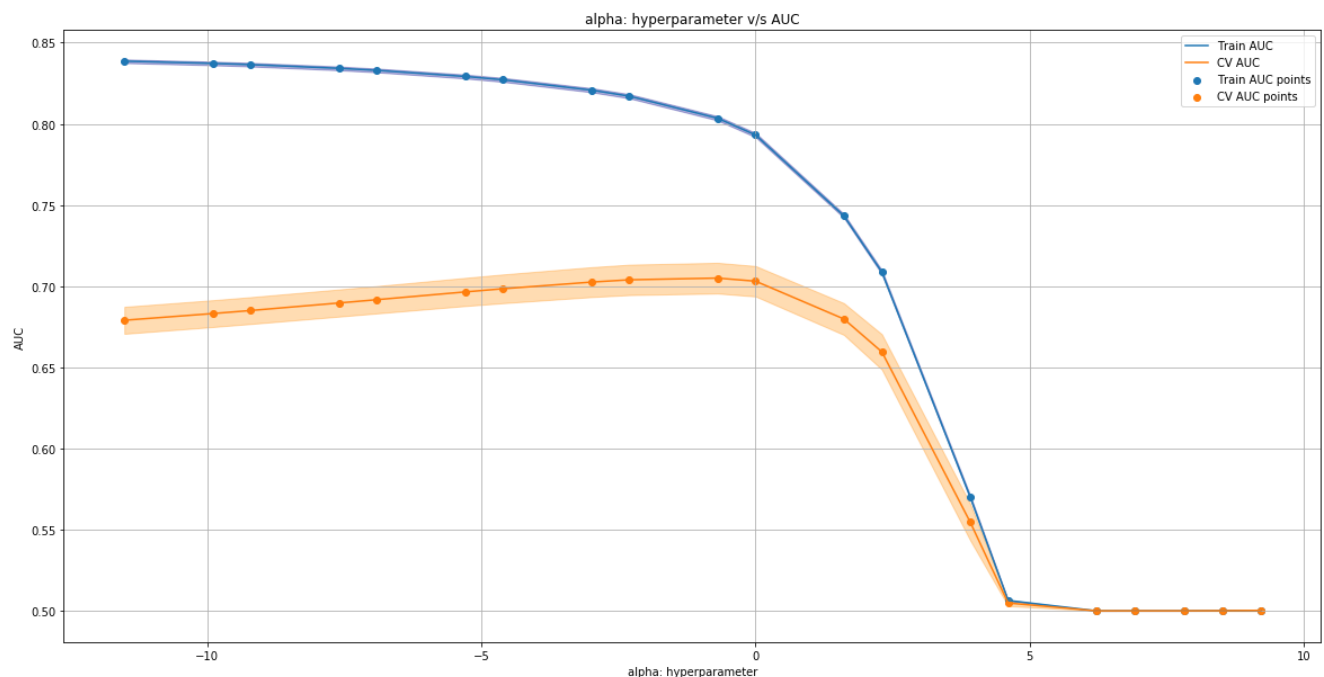
plt.figure(figsize=(20,10))

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

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

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

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

[illegible]

From here we can choose the right alpha

In [224]:

```
print(neighbors)
print(cv_auc)
```

```
[1e-05, 5e-05, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2
500, 5000, 10000]
[0.6872682350570067, 0.6902452690649797, 0.6916423387431463, 0.6950872015285351,
0.6966231430756712, 0.7001865405384595, 0.7016700816339806, 0.7048132082648746,
0.7059067475266466, 0.7068771697395503, 0.7053038688582115, 0.6835038735913277,
0.6628203161033231, 0.5676939396343061, 0.5051321626634883, 0.5, 0.5, 0.5, 0.5001367240907848, 0.5
007228407512122]
```

C) Train model using the best hyper-parameter value

In [225]:

```
best_alpha_1 = 1.0
```

In [226]:

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

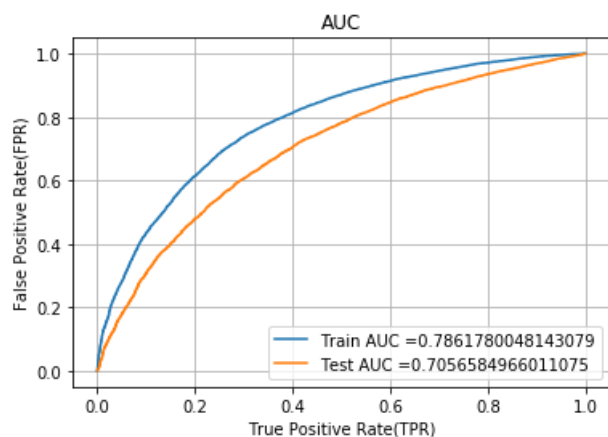
nb_bow = MultinomialNB(alpha = best_alpha_1)

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

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

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

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



D) Confusion Matrix

In [227]:

```
def predict(proba, threshold, fpr, tpr):  
  
    t = threshold[np.argmax(fpr*(1-tpr))]  
  
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high  
  
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))  
    predictions = []  
    for i in proba:  
        if i>=t:  
            predictions.append(1)  
        else:  
            predictions.append(0)  
    return predictions
```

Train Data

In [228]:

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

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.153
[[3713 3713]
 [5371 36244]]

In [229]:

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

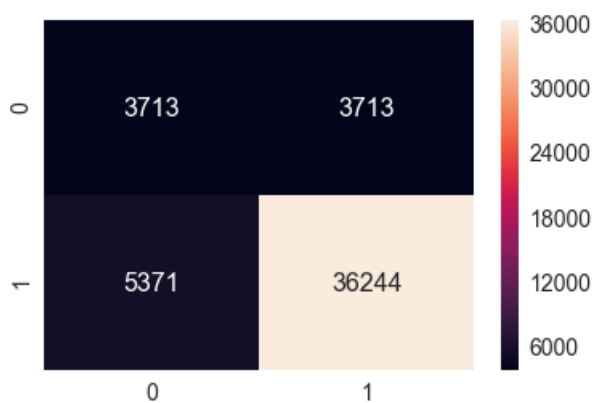
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.153

In [230]:

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

Out[230]:

<matplotlib.axes._subplots.AxesSubplot at 0x978b6ad208>



Test Data

In [231]:

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

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.628
[[ 2885  2574]
 [ 7280 23313]]
```

In [232]:

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

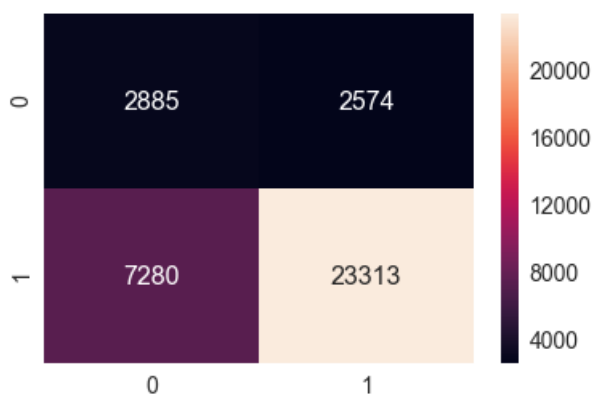
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.628

In [233]:

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

Out[233]:

<matplotlib.axes._subplots.AxesSubplot at 0x978c1ba240>



Set 2 : categorical, numerical features + project_title(TFIDF) + preprocessed_essay (TFIDF)

In [251]:

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

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

In [252]:

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

```
Final Data matrix
(49041, 14266) (49041,)
(24155, 14266) (24155,)
```

(36052, 14266) (36052,)

A) Finding Best alpha using random alpha values

In [253]:

```
# Importing libraries
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score, precision_score, recall_score

# Creating alpha values in the range from 10^-3 to 10^3
neighbors = [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]

# empty list that will hold cv scores
cv_scores = []
train_auc = []
cv_auc = []
log_alphas = []

# perform 10-fold cross validation
for k in tqdm(neighbors):
    bn = MultinomialNB(alpha = k)

    bn.fit(X_tr, y_train)
    y_train_pred = batch_predict(bn, X_tr)
    y_cv_pred = batch_predict(bn, X_cr)

    scores = cross_val_score(bn, X_tr, y_train, cv=10, scoring='accuracy', n_jobs=-1)
    cv_scores.append(scores.mean())

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates 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))

# determining best value of alpha
MSE = [1.0 - x for x in cv_scores]

optimal_alpha = neighbors[MSE.index(max(MSE))]
## optimal_alpha = neighbors[cv_scores.index(max(cv_scores))]
print('\nThe optimal value of alpha is %.3f.' % optimal_alpha)

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

100%|██| 20/20 [01:46<00:00, 5.36s/it]

The optimal value of alpha is 0.500.

100%|██| 20/20 [00:00<00:00, 19972.88it/s]

In [254]:

```
# ===== Multinomial Naive Bayes with alpha = optimal_alpha
# =====
# instantiate learning model alpha = optimal_alpha
bn_optimal = MultinomialNB(alpha = optimal_alpha)

# fitting the model
bn_optimal.fit(X_tr, y_train)

# predict the response
predictions = bn_optimal.predict(X_te)

# evaluate accuracy
acc = accuracy_score(y_test, predictions) * 100
print('\nThe Test Accuracy of the Bernoulli naive Bayes classifier for alpha = %.3f is %f%%' % (op
```

```

timal_alpha, acc))

# Variables that will be used for making table in Conclusion part of this assignment
bow_multinomial_alpha = optimal_alpha
bow_multinomial_train_acc = max(cv_scores)*100
bow_multinomial_test_acc = acc

```

The Test Accuracy of the Bernoulli naive Bayes classifier for $\alpha = 0.500$ is 84.380894%

In [237]:

```

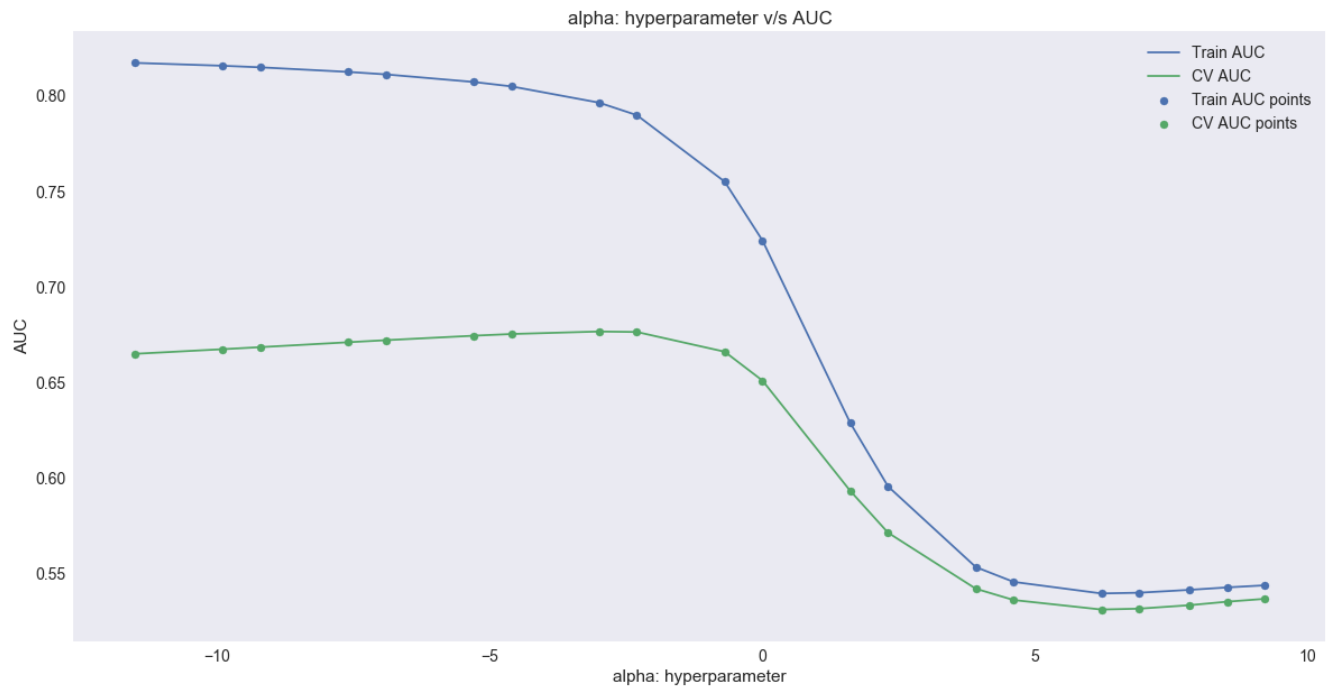
plt.figure(figsize=(20,10))

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

```



B) GridsearchCV using (10 fold cross validation)

In [99]:

```

nb = MultinomialNB()

parameters = {'alpha':[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]}

clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc')

clf.fit(X_tr, y_train)

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

```

In [100]:

```
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]
log_alphas = []

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

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

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

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

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()
```

alpha: hyperparameter v/s AUC

Legend:

- Train AUC (Blue line)
- CV AUC (Green line)
- Train AUC points (Blue dots)
- CV AUC points (Green dots)

log alpha: hyperparameter	Train AUC	CV AUC
-12	0.81	0.65
-10	0.81	0.65
-8	0.81	0.66
-6	0.80	0.66
-4	0.79	0.66
-2	0.78	0.67
0	0.72	0.64
2	0.63	0.59
4	0.55	0.54
6	0.54	0.54
8	0.54	0.54
10	0.54	0.54

In [238]:

```
[1e-05, 5e-05, 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]
[0.665079125082146, 0.6675008583775608, 0.6685817644347303, 0.6711374737320724,
0.6722251432894885, 0.6745632066173552, 0.6754510794426332, 0.6767709491682803,
0.6765156970702144, 0.6662074823629391, 0.6511052533240336, 0.593399842953392, 0.5715150237884443,
0.542204715117501, 0.536299252255694, 0.531316693681981, 0.531778806167768, 0.5335796181239715,
```

```
conf matr df train 2 = pd.DataFrame(confusion matrix(y train, predict(y train pred, tr thresholds,
```



```
train_fpr, train_fpr)), range(2),range(2))
```

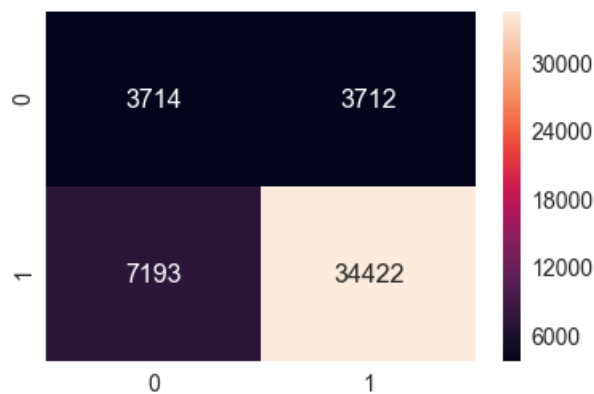
the maximum value of $tpr*(1-fpr)$ 0.2499999818661462 for threshold 0.791

In [243]:

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

Out[243]:

<matplotlib.axes._subplots.AxesSubplot at 0x978c2bfe80>



Test Data

In [244]:

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

Test confusion matrix

the maximum value of $tpr*(1-fpr)$ 0.24999999161092998 for threshold 0.844

```
[[ 2810  2649]
 [ 8751 21842]]
```

In [245]:

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

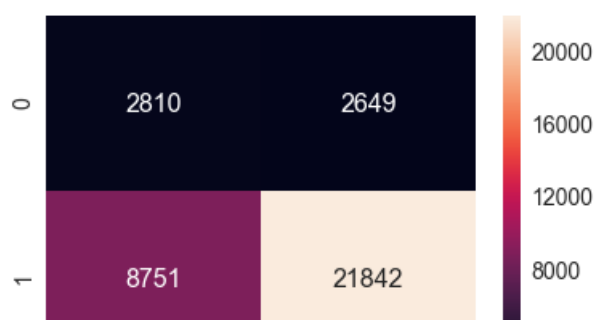
the maximum value of $tpr*(1-fpr)$ 0.24999999161092998 for threshold 0.844

In [246]:

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

Out[246]:

<matplotlib.axes._subplots.AxesSubplot at 0x978c17d860>





Select best 10 features of both Positive and negative class for both the sets of data

SET 1 : BOW

In [128]:

```
from scipy.sparse import hstack

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

In [129]:

```
from sklearn.naive_bayes import MultinomialNB

nb_bow = MultinomialNB(alpha = 0.5)

nb_bow.fit(X_tr, y_train)
```

Out[129]:

```
MultinomialNB(alpha=0.5, class_prior=None, fit_prior=True)
```

In [132]:

```
bow_features_names = []
```

In [133]:

```
for a in vectorizer_proj.get_feature_names() :
    bow_features_names.append(a)
```

In [134]:

```
for a in vectorizer_sub_proj.get_feature_names() :
    bow_features_names.append(a)
```

In [135]:

```
for a in vectorizer_states.get_feature_names() :
    bow_features_names.append(a)
```

In [136]:

```
for a in vectorizer_grade.get_feature_names() :
    bow_features_names.append(a)
```

In [137]:

```
for a in vectorizer_teacher.get_feature_names() :
    bow_features_names.append(a)
```

```
In [138]:
```

```
len(bow_features_names)
```

```
Out[138]:
```

```
100
```

```
In [139]:
```

```
bow_features_names.append("price")
```

```
In [140]:
```

```
bow_features_names.append("quantity")
```

```
In [141]:
```

```
bow_features_names.append("prev_proposed_projects")
```

```
In [142]:
```

```
bow_features_names.append("title_word_count")
```

```
In [143]:
```

```
bow_features_names.append("essay_word_count")
```

```
In [144]:
```

```
len(bow_features_names)
```

```
Out[144]:
```

```
105
```

```
In [145]:
```

```
for a in vectorizer_bow_title.get_feature_names() :  
    bow_features_names.append(a)
```

```
In [146]:
```

```
len(bow_features_names)
```

```
Out[146]:
```

```
2224
```

```
In [147]:
```

```
for a in vectorizer_bow_essay.get_feature_names() :  
    bow_features_names.append(a)
```

```
In [148]:
```

```
len(bow_features_names)
```

```
Out[148]:
```

```
14266
```

```
In [156]:
```

```
nb_bow.classes_
```

```
Out[156]:
```

```
array([0, 1], dtype=int64)
```

- It show that there are two classes [0 for Negative] and [1 for Positive]

Printing all Negative and Positive Features

```
In [155]:
```

```
# Now we can find log probabilities of different features for both the classes
class_features = nb_bow.feature_log_prob_

# row_0 is for 'negative' class and row_1 is for 'positive' class
negative_features = class_features[0]
positive_features = class_features[1]

# Sorting 'negative_features' and 'positive_features' in descending order using argsort() function
sorted_negative_features = np.argsort(negative_features)[::-1]
sorted_positive_features = np.argsort(positive_features)[::-1]

print("Top 20 Important Features and their log probabilities For Negative Class :\n\n")
for i in list(sorted_negative_features[0:20]):
    print("%s\t -->\t%f" %(bow_features_names[i],negative_features[i]))

print("\n\nTop 20 Important Features and their log probabilities For Positive Class :\n\n")
for i in list(sorted_positive_features[0:20]):
    print("%s\t -->\t%f" %(bow_features_names[i],positive_features[i]))
```

Top 20 Important Features and their log probabilities For Negative Class :

```
students --> -3.040735
school --> -4.119802
learning --> -4.450391
classroom --> -4.609650
not --> -4.801005
learn --> -4.813502
help --> -4.856444
price --> -4.958871
quantity --> -4.958871
essay_word_count --> -4.958871
title_word_count --> -4.958871
nannan --> -5.008419
many --> -5.035380
need --> -5.138329
work --> -5.190245
come --> -5.352190
prev_proposed_projects --> -5.353788
reading --> -5.384638
love --> -5.390011
able --> -5.423302
```

Top 20 Important Features and their log probabilities For Positive Class :

```
students --> -3.019299
school --> -4.164551
learning --> -4.520625
classroom --> -4.556703
not --> -4.822673
learn --> -4.864016
help --> -4.898187
price --> -5.010166
quantity --> -5.010166
title_word_count --> -5.010166
essay_word_count --> -5.010166
```

```
many --> -5.035795
nannan --> -5.054910
work --> -5.162802
need --> -5.164006
reading --> -5.179000
use --> -5.233937
prev_proposed_projects --> -5.319683
love --> -5.336858
able --> -5.348519
```

SET 2 : TFIDF

In [158]:

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

In [159]:

```
nb_tfidf = MultinomialNB(alpha = 0.1)

nb_tfidf.fit(X_tr, y_train)
```

Out[159]:

```
MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
```

In [160]:

```
tfidf_features_names = []
```

In [161]:

```
for a in vectorizer_proj.get_feature_names() :
    tfidf_features_names.append(a)
```

In [162]:

```
for a in vectorizer_sub_proj.get_feature_names() :
    tfidf_features_names.append(a)
```

In [163]:

```
for a in vectorizer_states.get_feature_names() :
    tfidf_features_names.append(a)
```

In [164]:

```
for a in vectorizer_grade.get_feature_names() :
    tfidf_features_names.append(a)
```

In [165]:

```
for a in vectorizer_teacher.get_feature_names() :
    tfidf_features_names.append(a)
```

In [166]:

In [166]:

```
len(tfidf_features_names)
```

Out[166]:

100

In [167]:

```
tfidf_features_names.append("price")
```

In [168]:

```
tfidf_features_names.append("quantity")
```

In [169]:

```
tfidf_features_names.append("prev_proposed_projects")
```

In [170]:

```
tfidf_features_names.append("title_word_count")
```

In [171]:

```
tfidf_features_names.append("essay_word_count")
```

In [172]:

```
for a in vectorizer_tfidf_titles.get_feature_names() :  
    tfidf_features_names.append(a)
```

In [173]:

```
for a in vectorizer_tfidf_essay.get_feature_names() :  
    tfidf_features_names.append(a)
```

In [174]:

```
len(tfidf_features_names)
```

Out[174]:

14266

Printing all Negative and Positive Features

In [175]:

```
# Now we can find log probabilities of different features for both the classes  
class_features = nb_tfidf.feature_log_prob_  
  
# row_0 is for 'negative' class and row_1 is for 'positive' class  
negative_features = class_features[0]  
positive_features = class_features[1]  
  
# Sorting 'negative_features' and 'positive_features' in descending order using argsort() function  
sorted_negative_features = np.argsort(negative_features)[::-1]  
sorted_positive_features = np.argsort(positive_features)[::-1]  
  
print("Top 20 Important Features and their log probabilities For Negative Class :\n\n")  
for i in list(sorted_negative_features[0:20]):  
    print("%s\t -->\t%f" % (tfidf_features_names[i], negative_features[i]))
```

```
print("\n\nTop 20 Important Features and their log probabilities For Positive Class :\n\n")
for i in list(sorted_positive_features[0:20]):
    print("%s\t -->\t%f" %(tfidf_features_names[i],positive_features[i]))
```

Top 20 Important Features and their log probabilities For Negative Class :

```
essay_word_count --> -2.952164
title_word_count --> -2.952164
price --> -2.952164
quantity --> -2.952164
prev_proposed_projects --> -3.347108
Literacy_Language --> -3.822832
Math_Science --> -3.878042
Literacy --> -4.303227
Mathematics --> -4.311581
Literature_Writing --> -4.637185
SpecialNeeds --> -4.938245
SpecialNeeds --> -4.938245
CA --> -4.973193
prides --> -4.990641
AppliedLearning --> -4.994763
Health_Sports --> -5.001013
AppliedSciences --> -5.109035
Music_Arts --> -5.273110
TX --> -5.446392
Health_Wellness --> -5.474512
```

Top 20 Important Features and their log probabilities For Positive Class :

```
title_word_count --> -2.957263
essay_word_count --> -2.957263
quantity --> -2.957263
price --> -2.957263
prev_proposed_projects --> -3.266783
Literacy_Language --> -3.675888
Math_Science --> -3.943461
Literacy --> -4.105198
Mathematics --> -4.325240
Literature_Writing --> -4.541070
CA --> -4.892784
prides --> -4.977209
Health_Sports --> -4.990092
SpecialNeeds --> -5.043221
SpecialNeeds --> -5.043221
AppliedLearning --> -5.190330
AppliedSciences --> -5.282058
Health_Wellness --> -5.305424
Music_Arts --> -5.329350
NY --> -5.638723
```

Pretty Table

In [257]:

```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha", "AUC Scores", "Accuracy"]

x.add_row(["BOW", "Naive Bayes", 1.0, 0.7, '74%'])
x.add_row(["TFIDF", "Naive Bayes", 0.5, 0.67, '84%'])

print(x)
```

+-----+-----+-----+-----+-----+

Vectorizer	Model	Alpha	AUC Scores	Accuracy
BOW	Naive Bayes	1.0	0.7	74%
TFIDF	Naive Bayes	0.5	0.67	84%

In [3]:

```
y = PrettyTable()
y.field_names = ["Vectorizer", "Model", "K in KNN", "AUC Scores"]

y.add_row(["BOW", "KNN", 91, 0.63])
y.add_row(["TFIDF", "KNN", 121, 0.62])

print(y)
```

Vectorizer	Model	K in KNN	AUC Scores
BOW	KNN	91	0.63
TFIDF	KNN	121	0.62

Conclusion

1. Naive bayes seems to function better than KNN for both Bag of Words model (BOW) as well as Term Frequency Inverse Document Frequency model (TFIDF).
2. Grid Search in Naive Bayes is very fast compared to KNN..
3. Appropriate alpha value is chosen using 10 Fold CV. when alpha=0-->OVERFIT and when alpha=large-->UNDERFIT