DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

Feature	Description
	• `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

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
```

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]:

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


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5

In [6]:

```
project_grade_category = []

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

In [7]:

```
project_grade_category[0:5]
```

Out[7]:

```
['Grades_PreK-2', 'Grades_6-8', 'Grades_6-8', 'Grades_PreK-2', 'Grades_PreK-2']
```

In [8]:

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

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

In [9]:

```
project_data["project_grade_category"] = project_grade_category
```

In [10]:

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

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_subject_ca
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Math & Science
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Special Needs
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Literacy & Language
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Applied Learning
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Literacy & Language

1.2 preprocessing of `project_subject_categories`

In [11]:

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

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

project_data['clean_categories'] = cat_list
```

```
project_data['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 [12]:

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

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

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

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

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

1.4 Introducing new feature "Number of Words in Title"

In [13]:

```
title_word_count = []
```

In [14]:

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

In [15]:

```
project_data["title_word_count"] = title_word_count
```

In [16]:

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

Out[16]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	projec
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	I have fortuna to use ...
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagin 9 years You're th...
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Mobile Learning with a Mobile Listening Center	Having 24 stu comes diver...
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recer article giving
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My stu crave they ea obstac

1.5 Combine 4 Project essays into 1 Essay

In [17]:

```
# 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)
```

1.6 Introducing new feature "Number of Words in Essay"

In [18]:

```
essay_word_count = []
```

In [19]:

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

In [20]:

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

In [21]:

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

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	projec
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	I have fortuna to use ...
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagin 9 years You're th...
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Mobile Learning with a Mobile Listening Center	Having 24 stu comes diver...
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recer article giving
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My stu crave they ea obstac

1.7 Test - Train Split

In [22]:

```
# train test split

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)
```

In [23]:

```
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)
```

1.8 Text preprocessing

In [24]:

```
# 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)
```

I have been fortunate enough to use the Fairy Tale STEM kits in my classroom as well as the STEM j ournals, which my students really enjoyed. I would love to implement more of the Lakeshore STEM k its in my classroom for the next school year as they provide excellent and engaging STEM lessons.My students come from a variety of backgrounds, including language and socioeconomic statu s. Many of them don't have a lot of experience in science and engineering and these kits give me the materials to provide these exciting opportunities for my students.Each month I try to do several science or STEM/STEAM projects. I would use the kits and robot to help guide my science i nstruction in engaging and meaningful ways. I can adapt the kits to my current language arts paci ng guide where we already teach some of the material in the kits like tall tales (Paul Bunyan) or Johnny Appleseed. The following units will be taught in the next school year where I will implement these kits: magnets, motion, sink vs. float, robots. I often get to these units and don 't know If I am teaching the right way or using the right materials. The kits will give me additional ideas, strategies, and lessons to prepare my students in science.It is challenging to d evelop high quality science activities. These kits give me the materials I need to provide my students with science activities that will go along with the curriculum in my classroom. Although I have some things (like magnets) in my classroom, I don't know how to use them effectively. The kits will provide me with the right amount of materials and show me how to use them in an appropriate way.

==========

I teach high school English to students with learning and behavioral disabilities. My students all vary in their ability level. However, the ultimate goal is to increase all students literacy level s. This includes their reading, writing, and communication levels.I teach a really dynamic group o f students. However, my students face a lot of challenges. My students all live in poverty and in a dangerous neighborhood. Despite these challenges, I have students who have the the desire to def eat these challenges. My students all have learning disabilities and currently all are performing below grade level. My students are visual learners and will benefit from a classroom that fulfills their preferred learning style.The materials I am requesting will allow my students to be prepared for the classroom with the necessary supplies. Too often I am challenged with students who come t o school unprepared for class due to economic challenges. I want my students to be able to focus on learning and not how they will be able to get school supplies. The supplies will last all year . Students will be able to complete written assignments and maintain a classroom journal. The ch art paper will be used to make learning more visual in class and to create posters to aid students in their learning. The students have access to a classroom printer. The toner will be used to pr int student work that is completed on the classroom Chromebooks.I want to try and remove all barri ers for the students learning and create opportunities for learning. One of the biggest barriers i s the students not having the resources to get pens, paper, and folders. My students will be able to increase their literacy skills because of this project.

==========

\"Life moves pretty fast. If you don't stop and look around once in awhile, you could miss it.\" from the movie, Ferris Bueller's Day Off. Think back...what do you remember about your grandparents? How amazing would it be to be able to flip through a book to see a day in their lives?My second graders are voracious readers! They love to read both fiction and nonfiction books . Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson. They also love to read about insects, space and plants. My students are hungry bookworms! My stude nts are eager to learn and read about the world around them. My kids love to be at school and are like little sponges absorbing everything around them. Their parents work long hours and usually do not see their children. My students are usually cared for by their grandparents or a family friend. Most of my students do not have someone who speaks English at home. Thus it is difficult f or my students to acquire language.Now think forward... wouldn't it mean a lot to your kids, nieces or nephews or grandchildren, to be able to see a day in your life today 30 years from now? Memories are so precious to us and being able to share these memories with future generations will be a rewarding experience. As part of our social studies curriculum, students will be learning ab out changes over time. Students will be studying photos to learn about how their community has ch anged over time. In particular, we will look at photos to study how the land, buildings, clothing, and schools have changed over time. As a culminating activity, my students will capture a slice of their history and preserve it through scrap booking. Key important events in their young lives will be documented with the date, location, and names. Students will be using photos from home and from school to create their second grade memories. Their scrap books will preserve their unique stories for future generations to enjoy.Your donation to this project will provide my second graders with an opportunity to learn about social studies in a fun and creative manner. Th rough their scrapbooks, children will share their story with others and have a historical document for the rest of their lives.

==========

\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the bi ggest enthusiasm for learning. My students learn in many different ways using all of our senses an d multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nSt udents in my class come from a variety of different backgrounds which makes for wonderful sharing of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su ccessful learners which can be seen through collaborative student project based learning in and ou t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many

different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it's healthy for their bodies. This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroo
m garden in the spring. We will also create our own cookbooks to be printed and shared with famili
es. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
==
My classroom consists of twenty-two amazing sixth graders from different cultures and backgrounds.
They are a social bunch who enjoy working in partners and working with groups. They are hard-worki
ng and eager to head to middle school next year. My job is to get them ready to make this
transition and make it as smooth as possible. In order to do this, my students need to come to
school every day and feel safe and ready to learn. Because they are getting ready to head to
middle school, I give them lots of choice- choice on where to sit and work, the order to complete
assignments, choice of projects, etc. Part of the students feeling safe is the ability for them to
come into a welcoming, encouraging environment. My room is colorful and the atmosphere is casual.
I want them to take ownership of the classroom because we ALL share it together. Because my time w
ith them is limited, I want to ensure they get the most of this time and enjoy it to the best of t
heir abilities.Currently, we have twenty-two desks of differing sizes, yet the desks are similar t
o the ones the students will use in middle school. We also have a kidney table with crates for sea
ting. I allow my students to choose their own spots while they are working independently or in
groups. More often than not, most of them move out of their desks and onto the crates. Believe it
or not, this has proven to be more successful than making them stay at their desks! It is because
of this that I am looking toward the “Flexible Seating” option for my classroom.\r\n The students
look forward to their work time so they can move around the room. I would like to get rid of the c
onstricting desks and move toward more “fun” seating options. I am requesting various seating so m
y students have more options to sit. Currently, I have a stool and a papasan chair I inherited fro
m the previous sixth-grade teacher as well as five milk crate seats I made, but I would like to gi
ve them more options and reduce the competition for the “good seats”. I am also requesting two rug
s as not only more seating options but to make the classroom more welcoming and appealing. In orde
r for my students to be able to write and complete work without desks, I am requesting a class set
of clipboards. Finally, due to curriculum that requires groups to work together, I am requesting t
ables that we can fold up when we are not using them to leave more room for our flexible seating o
ptions.\r\nI know that with more seating options, they will be that much more excited about coming
to school! Thank you for your support in making my classroom one students will remember
forever!nannan
==

In [25]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [26]:

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

\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the b
iggest enthusiasm for learning. My students learn in many different ways using all of our senses a
nd multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nS

tudents in my class come from a variety of different backgrounds which makes for wonderful sharing of experiences and cultures, including Native Americans.\r\nOur school is a caring community of successful learners which can be seen through collaborative student project based learning in and out of the classroom. Kindergarteners in my class love to work with hands-on materials and have many different opportunities to practice a skill before it is mastered. Having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the perfect place to learn about agriculture and nutrition. My students love to role play in our pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try cooking with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we learn important math and writing concepts while cooking delicious healthy food for snack time. My students will have a grounded appreciation for the work that went into making the food and knowledge of where the ingredients came from as well as how it is healthy for their bodies. This project would expand our learning of nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroom garden in the spring. We will also create our own cookbooks to be printed and shared with families. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healthy cooking.nannan
==========

In [27]:

```
# \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)
```

 A person is a person, no matter how small. (Dr.Seuss) I teach the smallest students with the biggest enthusiasm for learning. My students learn in many different ways using all of our senses and multiple intelligences. I use a wide range of techniques to help all my students succeed. Students in my class come from a variety of different backgrounds which makes for wonderful sharing of experiences and cultures, including Native Americans. Our school is a caring community of successful learners which can be seen through collaborative student project based learning in and out of the classroom. Kindergarteners in my class love to work with hands-on materials and have many different opportunities to practice a skill before it is mastered. Having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the perfect place to learn about agriculture and nutrition. My students love to role play in our pretend kitchen in the early childhood classroom. I have had several kids ask me, Can we try cooking with REAL food? I will take their idea and create Common Core Cooking Lessons where we learn important math and writing concepts while cooking delicious healthy food for snack time. My students will have a grounded appreciation for the work that went into making the food and knowledge of where the ingredients came from as well as how it is healthy for their bodies. This project would expand our learning of nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroom garden in the spring. We will also create our own cookbooks to be printed and shared with families.   Students will gain math and literature skills as well as a life long enjoyment for healthy cooking.nannan

In [28]:

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

 A person is a person no matter how small Dr Seuss I teach the smallest students with the biggest enthusiasm for learning My students learn in many different ways using all of our senses and multiple intelligences I use a wide range of techniques to help all my students succeed Students in my class come from a variety of different backgrounds which makes for wonderful sharing of experiences and cultures including Native Americans Our school is a caring community of successful learners which can be seen through collaborative student project based learning in and out of the classroom Kindergarteners in my class love to work with hands on materials and have many different opportunities to practice a skill before it is mastered Having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curriculum Montana is the perfect place to learn about agriculture and nutrition My students love to role play in our pretend kitchen in the early childhood classroom I have had several kids ask me Can we try cooking with REAL food I will take their idea and create Common Core Cooking Lessons where we learn important math and writing concepts while cooking delicious healthy food for snack time My students will have a grounded appreciation for the work that went into making the food and knowledge of where the ingredients came from as well as how it is healthy for their bodies This project would expand our learning of nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce make our own bread and mix up healthy plants from our classroom garden in the spring We will also create our own cookbooks to be printed and shared with families Students will gain math and literature skills as well as a life long enjoyment for healthy cooking nannan

In [29]:

```
# 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', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

1.8.1 Preprocessed Train data (Text)

In [30]:

```
# 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('\\"', ' ')
    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 [00:28<00:00, 1745.55it/s]
```

In [31]:

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

Out[31]:

```
'whole school k 1 building also title 1 building 75 80 students receiving free reduced lunch stude
nts attend school live within district boundaries well districts neighboring communities school bu
ilding full boundless energy miles smiles giggles well teaching future teachers engineers doctors
writers dreamers time children come school eager learn excited new learning adventure awaits given
day many families daily struggles ranging newly immigrated lack permament housing well maintaining
employment however heart families concern children offered better life listening center headphones
aid helping children pay close attention words stories read via cd story write wipe boards help wh
ole group instruction individual answer needed given materials certainly well received one since c
hild able experience materials daily children love listen stories read helps language development
students either speak english first second language children also love write answer share work who
```

```
le group instruction time nannan'
```

1.8.2 Preprocessed Test data (Text)

In [32]:

```
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 [00:20<00:00, 1731.23it/s]
```

In [33]:

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

Out[33]:

```
'students school stem program inquisitive group youngsters love learning various things science te
chnology engineering mathematics comes challenge work teams design engineering tinker way best pos
sible solution stem learners set long term goals become best future stem professionals many learne
rs go amazing things school beyond going amazing places computer science exploding right teaching
students think computational future job skills every student need stem class work real world
programming languages javascript python ruby kids love learning computer science would ideal could
1 1 student computer ratio students learn coding chromebooks come enough chromebook students would
able workstation order complete coding adventures skills like futures looking bright nannan'
```

1.8.3 Preprocessed Cross Validation data (Text)

In [34]:

```
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 [00:13<00:00, 1740.36it/s]
```

In [35]:

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

Out[35]:

```
'teach 1st grade k 5 district california students future contributing members society change world
population students school come variety diverse backgrounds many students come multi generational
homes immigrant families students english language learners one third students low income families
students perseverant curious eager learn everyday students innovative knowledgeable creative day s
tudents make connections learning daily lives students taught independent explorers try best never
give goal students tools need help diverse needs students best learners teach 1st grade k 5 distri
ct california students future contributing members society change world population students school
come variety diverse backgrounds many students come multi generational homes immigrant families st
```

udents english language learners one third students low income families students perseverant curio us eager learn everyday students innovative knowledgeable creative day students make connections l earning daily lives students taught independent explorers try best never give goal students tools need help diverse needs students best learners nannan'

1.9 Preprocessing of Project_title

In [36]:

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

```
Engineering STEAM into the Primary Classroom
==================================================
Building Blocks for Learning
==================================================
Empowering Students Through Art:Learning About Then and Now
==================================================
Health Nutritional Cooking in Kindergarten
==================================================
```

1.9.1 Preprocessing of Project Title for Train data

In [37]:

```
preprocessed_titles_train = []

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

```
100%|██████████| 49041/49041 [00:02<00:00, 24075.56it/s]
```

In [38]:

```
preprocessed_titles_train[1000]
```

Out[38]:

```
'listening teaching building oh my'
```

1.9.2 Preprocessing of Project Title for Test data

In [39]:

```
preprocessed_titles_test = []

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

```
100%|██████████| 36052/36052 [00:01<00:00, 23508.12it/s]
```

In [40]:

```
preprocessed_titles_test[1000]
```

Out[40]:

```
'chromebooks stem education'
```

1.9.3 Preprocessing of Project Title for Cross Validation data

In [41]:

```
preprocessed_titles_cv = []

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

```
100%|██████████| 24155/24155 [00:01<00:00, 24145.54it/s]
```

In [42]:

```
preprocessed_titles_cv[1000]
```

Out[42]:

```
'for love reading'
```

1.10 Preparing data for models

In [43]:

```
project_data.columns
```

Out[43]:

```
Index([u'Unnamed: 0', u'id', u'teacher_id', u'teacher_prefix', u'school_state',
       u'Date', u'project_title', u'project_essay_1', u'project_essay_2',
       u'project_essay_3', u'project_essay_4', u'project_resource_summary',
       u'teacher_number_of_previously_posted_projects', u'project_is_approved',
       u'project_grade_category', u'clean_categories', u'clean_subcategories',
       u'title_word_count', u'essay', u'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

- 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 Encode - Clean Categories of Projects

In [175]:

```
# 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)
```

```
['SpecialNeeds', 'Music_Arts', 'Math_Science', 'Health_Sports', 'Care_Hunger',
'Literacy_Language', 'AppliedLearning', 'History_Civics', 'Warmth']
('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 Encode - Clean Sub-Categories of Projects

In [176]:

```
# 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)
```

```
['Health_Wellness', 'Literature_Writing', 'CommunityService', 'Care_Hunger', 'AppliedSciences', 'S
ocialSciences', 'Other', 'Music', 'Mathematics', 'Warmth', 'EnvironmentalScience',
'ForeignLanguages', 'NutritionEducation', 'TeamSports', 'Extracurricular', 'Literacy',
'SpecialNeeds', 'PerformingArts', 'Health_LifeScience', 'Economics', 'ParentInvolvement',
'EarlyDevelopment', 'FinancialLiteracy', 'ESL', 'Civics_Government', 'CharacterEducation',
'History_Geography', 'VisualArts', 'College_CareerPrep', 'Gym_Fitness']
('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 Encode - School States

In [46]:

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

In [47]:

```
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 [177]:

```
## 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)
```

```
['WA', 'DE', 'DC', 'WI', 'WV', 'HI', 'FL', 'WY', 'NH', 'NJ', 'NM', 'TX', 'LA', 'NC', 'ND', 'NE', 'T
N', 'NY', 'PA', 'RI', 'NV', 'VA', 'CO', 'AK', 'AL', 'AR', 'VT', 'IL', 'GA', 'IN', 'IA', 'MA', 'AZ',
'CA', 'ID', 'CT', 'ME', 'MD', 'OK', 'OH', 'UT', 'MO', 'MN', 'MI', 'KS', 'MT', 'MS', 'SC', 'KY', 'OR
', 'SD']
('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 Encode - Project Grade Category

In [49]:

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

In [50]:

```
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 [178]:

```
## 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_6-8', 'Grades_9-12', 'Grades_PreK-2', 'Grades_3-5']
('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 Encode - Teacher Prefix

In [52]:

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

In [53]:

```
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 [179]:

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

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

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

print(vectorizer_teacher.get_feature_names())

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

```
['nan', 'Mrs.', 'Ms.', 'Mr.', 'Dr.', 'Teacher']
('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

A) Bag of words (BOW)

Bag of words - Train Data - Essays

In [165]:

```
# 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, 12060))
```

Bag of words - Test Data - Essays

In [166]:

```
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, 12060))
```

Bag of words - Cross Validation Data - Essays

In [167]:

```
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, 12060))
```

Bag of words - Train Data - Titles

In [168]:

```
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, 2079))
```

Bag of words - Test Data - Titles

In [169]:

```
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, 2079))
```

Bag of words - Cross Validation Data - Titles

In [170]:

In [170]:

```
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, 2079))
```

B) TFIDF vectorizer

TFIDF - Train Data - Essays

In [313]:

```
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, 12060))
```

TFIDF - Test Data - Essays

In [314]:

```
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, 12060))
```

TFIDF - Cross Validation Data - Essays

In [315]:

```
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, 12060))
```

TFIDF - Train Data - Titles

In [316]:

```
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, 2079))
```

TFIDF - Test Data - Titles

In [317]:

```
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, 2079))
```

TFIDF - Cross Validation Data - Titles

In [318]:

```
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, 2079))
```

1.12 Vectorizing Numerical features

A) Price

In [67]:

```
# 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[67]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [68]:

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

In [69]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

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

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

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

B) Quantity

In [70]:

```
normalizer = Normalizer()

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

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

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

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

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

C) Number of Projects previously proposed by Teacher

In [71]:

```
normalizer = Normalizer()

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

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

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

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

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

D) Title word Count

In [72]:

```
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)
print("="*100)
```

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

E) Essay word Count

In [181]:

```
normalizer_ess_count = Normalizer()

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

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

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

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

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
2. **The hyper paramter tuning(find best Alpha)**
   - Find the best hyper parameter which will give the maximum AUC value
   - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
   - Find the best hyper paramter using k-fold cross validation or simple cross validation data
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning
3. **Feature importance**

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

4. **Representation of results**

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot 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 [74]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
## https://github.com/harrismohammed/DonorsChoose.org---
NaiveBayes/blob/master/DonorsChoose_NB.ipynb
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
title_bow_train,
               text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test,
               quantity_test, prev_projects_test, title_word_count_test, essay_word_count_test, tit
le_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 [75]:

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

```
Final Data matrix
((49041, 14244), (49041,))
((24155, 14244), (24155,))
((36052, 14244), (36052,))
====================================================================================================
```

In [76]:

```
def batch_predict(clf, data):
```

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your tr_loop will be 49041 - 49041%1000 = 49000
    # in this for loop we will iterate until 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
```

A) Random alpha values

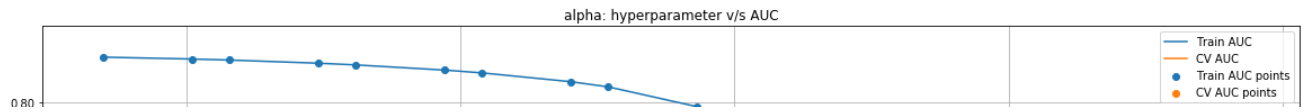
In [77]:

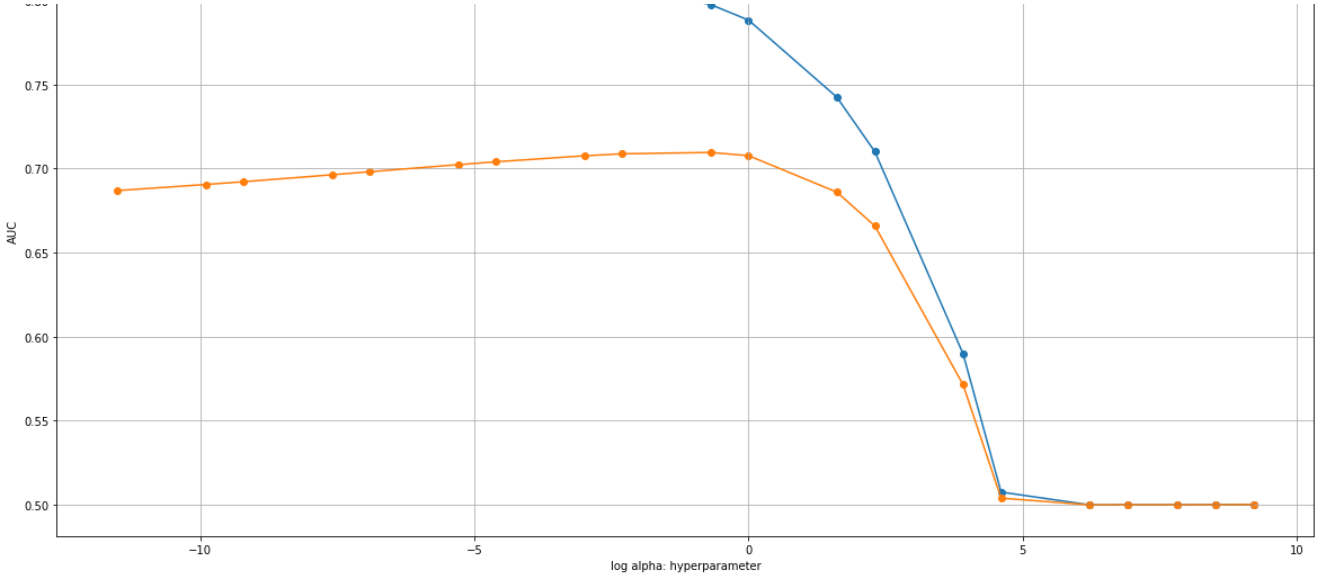
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

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, 5
00, 1000, 2500, 5000, 10000]

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

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


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

```
100%|██████████| 20/20 [00:03<00:00,  6.71it/s]
100%|██████████| 20/20 [00:00<00:00, 7241.55it/s]
```

In [78]:

```
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 AUC values on the Y axis.

3. We notice that very low or very high values of Alpha seem to be ineffective while developing the required model.

B) Gridsearch-cv using cv = 10 (K fold cross validation)

In [79]:

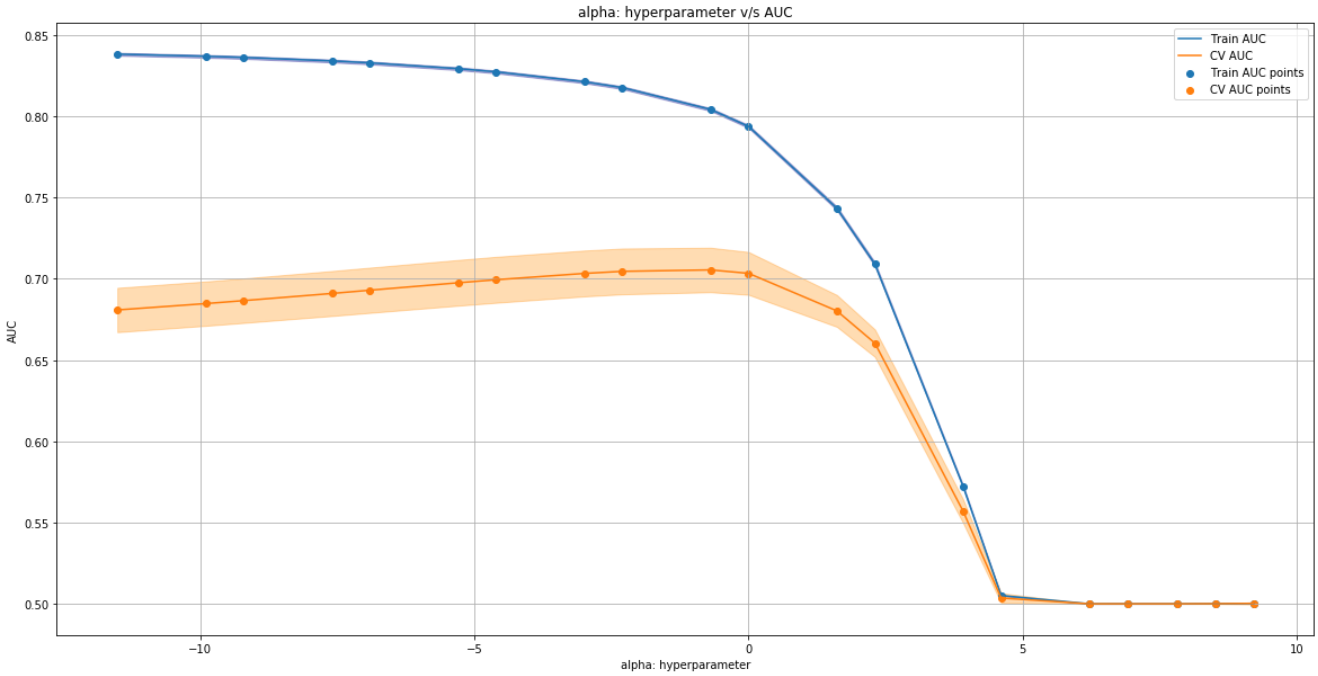
```
##https://github.com/harrismohammed/DonorsChoose.org---
NaiveBayes/blob/master/DonorsChoose_NB.ipynb
from sklearn.model_selection import GridSearchCV

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',return_train_score=True)

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 [80]:

```
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, 5
00, 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,col
or='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='darkoran
ge')

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


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

```
100%|██████████| 20/20 [00:00<00:00, 7462.51it/s]
```



Summary of Alpha values for BOW model :

Alpha values ranging from 0.00001 to 10000.0 was taken and the following results were obtained :

1. 0.00001 as alpha values seemed to work very well on train data and the model seems to not work that efficiently on cross validation data.
2. Values closer to 1.0 works pretty well both on Train data and Cross Validation data.
3. Values more than 1.0 also doesnt seem to be effective on both Train and Cross Validation data.

0.5 as alpha value was chosen. Even 1.0 resulted in an almost similar result.

C) Train model using the best hyper-parameter value

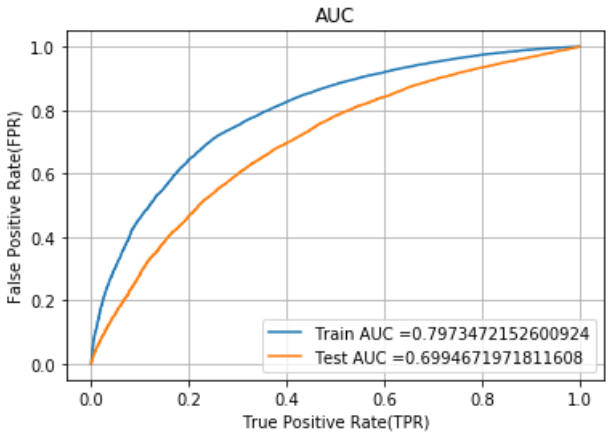
In [81]:

```
best_k_1 = 0.5
```

In [82]:

```
# 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_k_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 [83]:

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

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

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

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

Train Data

In [84]:

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

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

Train confusion matrix
('the maximum value of tpr*(1-fpr)', 0.25, 'for threshold', 0.122)
[[ 2712  2712]
```

```
[[ 3713  3713]
 [ 4950 36665]]
```

In [85]:

```
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))
```

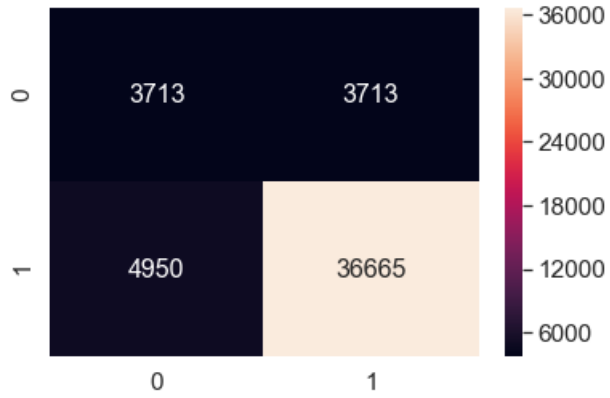
```
('the maximum value of tpr*(1-fpr)', 0.25, 'for threshold', 0.122)
```

In [113]:

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

Out[113]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a218824d0>
```



Test Data

In [86]:

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

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

Test confusion matrix
('the maximum value of tpr*(1-fpr)', 0.24999999161092998, 'for threshold', 0.581)
[[ 2798  2661]
 [ 6997 23596]]
```

In [87]:

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

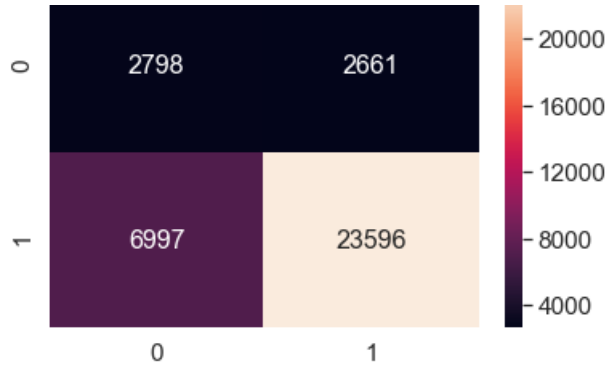
```
('the maximum value of tpr*(1-fpr)', 0.24999999161092998, 'for threshold', 0.581)
```

In [114]:

```
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[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a26ddd190>
```



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

In [88]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
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, tex
t_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 [89]:

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

```
Final Data matrix
((49041, 14244), (49041,))
((24155, 14244), (24155,))
((36052, 14244), (36052,))
====================================================================================================
```

A) Random alpha values

In [90]:

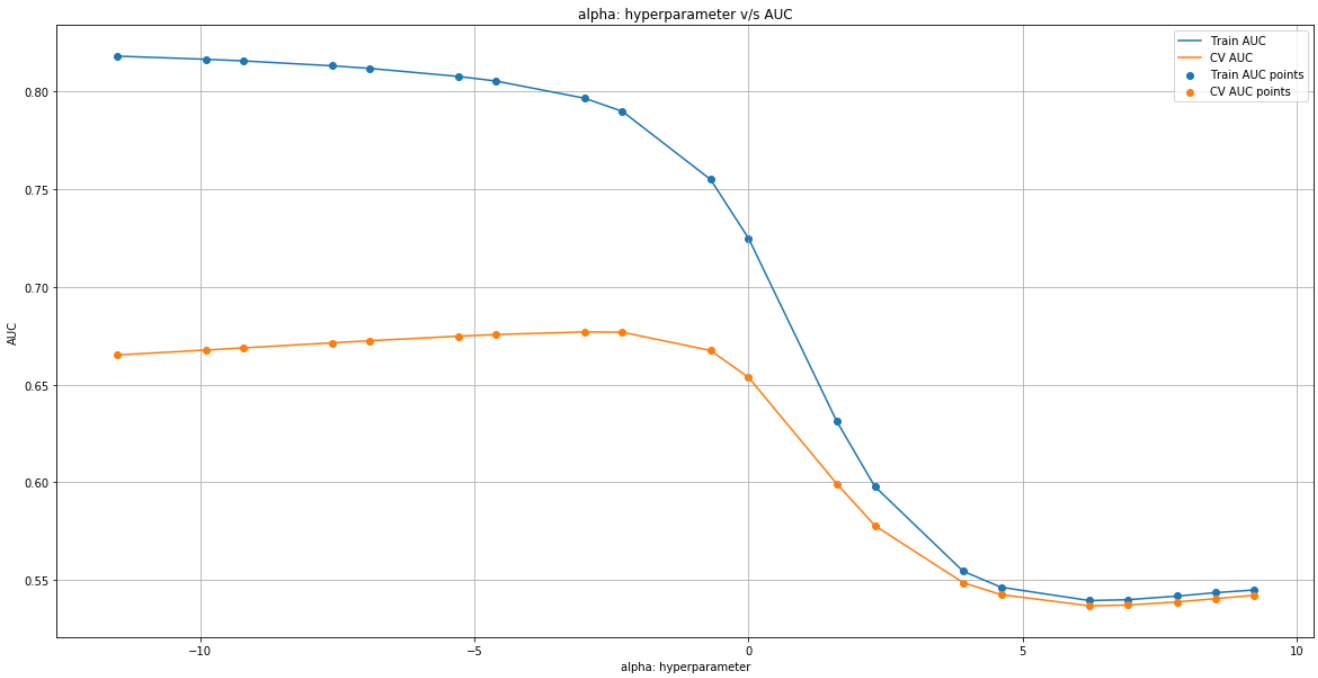
```
train_auc = []
cv_auc = []
log_alphas =[]

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, 5
00, 1000, 2500, 5000, 10000]
```

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

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


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

```
100%|██████████| 20/20 [00:03<00:00,  6.37it/s]
100%|██████████| 20/20 [00:00<00:00, 7191.26it/s]
```

In [91]:

```
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) Gridsearch-cv using cv = 10 (K fold cross validation)

In [92]:

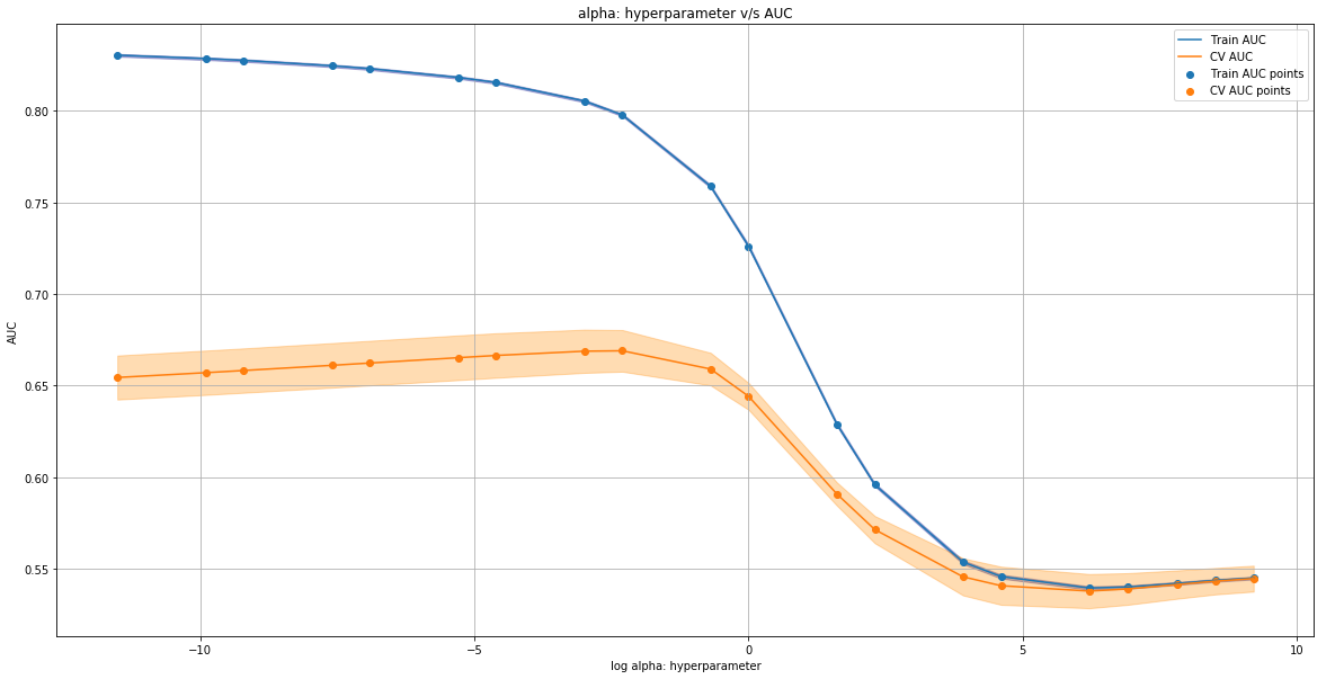
```
nb = MultinomialNB()

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',return_train_score=True)

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 [93]:

```
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, 5
00, 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,col
or='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='darkoran
ge')

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


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

```
100%|██████████| 20/20 [00:00<00:00, 7396.71it/s]
```



Summary of Alpha values for TFIDF model :

Alpha values ranging from 0.00001 to 10000.0 was taken and the following results were obtained :

1. 0.00001 as alpha values seemed to work very well on train data and the model seems to not work that efficiently on cross validation data.
2. Values closer to 0.1 works pretty well both on Train data and Cross Validation data.
3. Values more than 0.1 also doesnt seem to be effective on both Train and Cross Validation data.

0.1 as alpha value was chosen. Alpha values between 0.05 to 0.18 seemed to work better than the rest of the values

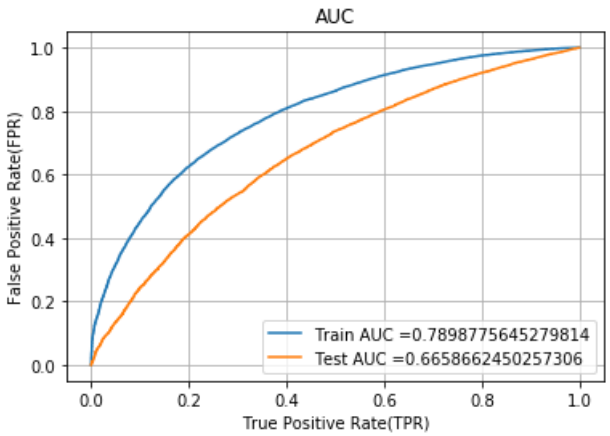
C) Train model using the best hyper-parameter value

In [94]:

```
best_k_2 = 0.1
```

In [95]:

```
nb_tfidf = MultinomialNB(alpha = best_k_2)

nb_tfidf.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_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_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

Train Data

In [96]:

```
print("="*100)
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', 0.76)
[[ 3713  3713]
 [ 5674 35941]]
```

In [97]:

```
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))
```

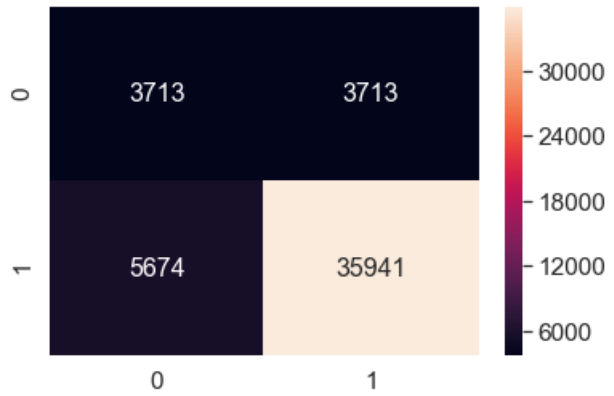
```
('the maximum value of tpr*(1-fpr)', 0.25, 'for threshold', 0.76)
```

In [115]:

```
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[115]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a292c7590>
```



Test Data

In [98]:

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

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

Test confusion matrix
('the maximum value of tpr*(1-fpr)', 0.24999999161092998, 'for threshold', 0.819)
[[ 2625  2834]
 [ 7669 22924]]
```

In [99]:

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

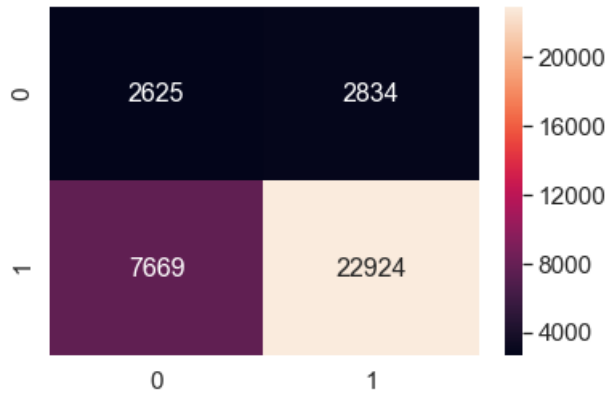
```
('the maximum value of tpr*(1-fpr)', 0.24999999161092998, 'for threshold', 0.819)
```

In [116]:

```
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[116]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a2e6ee910>
```



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

SET 1 : BOW

In [100]:

```
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
title_bow_train,
               text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test,
               quantity_test, prev_projects_test, title_word_count_test, essay_word_count_test, tit
le_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 [101]:

```
## https://github.com/harrismohammed/DonorsChoose.org---
NaiveBayes/blob/master/DonorsChoose_NB.ipynb
omialNB(alpha = 0.5)

nb_bow.fit(X_tr, y_train)
```

Out[101]:

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

In [131]:

```
bow_features_probs = {}

for a in range(14244) :

    bow_features_probs[a] = nb_bow.feature_log_prob_[0,a]
```

In [171]:

```
len(bow_features_probs.values())
```

Out[171]:

14244

In [222]:

```
bow_features_names = []
```

In [223]:

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

In [224]:

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

In [225]:

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

In [226]:

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

In [227]:

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

In [228]:

```
len(bow_features_names)
```

Out[228]:

100

In [229]:

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

In [230]:

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

In [231]:

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

In [232]:

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

In [233]:

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

In [234]:

```
len(bow_features_names)
```

Out[234]:

105

In [235]:

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

In [236]:

```
len(bow_features_names)
```

Out[236]:

2184

In [237]:

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

In [238]:

```
len(bow_features_names)
```

Out[238]:

14244

In [294]:

```
final_bow_features = pd.DataFrame({'feature_prob_estimates' : bow_features_probs.values(),
'feature_names' : bow_features_names})
```

In [295]:

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

25 Negative features from BOW model

In [296]:

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

Out[296]:

	feature_names	feature_prob_estimates
1962	then	-14.566227
11740	selfies	-14.566227
9809	palettes	-14.566227
9129	mondays	-14.566227
719	epic	-14.566227
720	equal	-14.566227
5111	denham	-14.566227
8535	lip	-14.566227

	feature_names	feature_prob_estimates
8525	lip	-14.566227
1775	snug	-14.566227
1774	snow	-14.566227
8526	lips	-14.566227
6279	extrinsic	-14.566227
6280	exuberance	-14.566227
9492	nutri	-14.566227
12726	surfing	-14.566227
733	everybody	-14.566227
8320	lapboards	-14.566227
3439	blades	-14.566227
9127	monarch	-14.566227
12097	sneakers	-14.566227
1766	sky	-14.566227
739	exam	-14.566227
9442	notate	-14.566227
11752	senegal	-14.566227
9808	palette	-14.566227

In [287]:

```
bow_features_probs_positive = {}

for a in range(14244) :

    bow_features_probs_positive[a] = nb_bow.feature_log_prob_[1,a]
```

In [288]:

```
final_bow_features_pos = pd.DataFrame({'feature_prob_estimates' : bow_features_probs_positive.valu
es(), 'feature_names' : bow_features_names})
```

In [289]:

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

25 Positive features from BOW model

In [292]:

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

Out[292]:

	feature_names	feature_prob_estimates
97	Mr.	-16.340276
95	Mrs.	-16.340276
94	nan	-16.340276
93	Grades_3-5	-16.340276
92	Grades_PreK-2	-16.340276
91	Grades_9-12	-16.340276
90	Grades_6-8	-16.340276

	feature_names	feature_prob_estimates
98	Dr.	-16.340276
96	Ms.	-16.340276
1783	soil	-14.143052
13546	unify	-14.143052
10365	praising	-14.143052
614	dog	-13.942381
4362	committing	-13.942381
4037	charitable	-13.942381
10425	preservation	-13.942381
10045	pervasive	-13.942381
3353	belorussia	-13.942381
9821	panthers	-13.942381
3614	breed	-13.775327
8074	ivy	-13.775327
9475	nudge	-13.775327
1183	leveling	-13.775327
12098	sneaky	-13.775327
13532	unfilled	-13.775327

SET 2 : TFIDF

In [297]:

```
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
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, tex
t_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 [298]:

```
nb_tfidf = MultinomialNB(alpha = 0.1)

nb_tfidf.fit(X_tr, y_train)
```

Out[298]:

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

In [322]:

```
tfidf_features_probs_neg = {}
```

```
for a in range(14244) :

    tfidf_features_probs_neg[a] = nb_tfidf.feature_log_prob_[0,a]
```

In [323]:

```
len(tfidf_features_probs_neg)
```

Out[323]:

```
14244
```

In [301]:

```
tfidf_features_names = []
```

In [302]:

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

In [303]:

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

In [304]:

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

In [305]:

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

In [306]:

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

In [307]:

```
len(tfidf_features_names)
```

Out[307]:

```
100
```

In [308]:

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

In [309]:

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

In [310]:

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

In [311]:

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

In [312]:

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

In [319]:

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

In [320]:

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

In [321]:

```
len(tfidf_features_names)
```

Out[321]:

```
14244
```

In [324]:

```
final_tfidf_features_neg = pd.DataFrame({'feature_prob_estimates' :
tfidf_features_probs_neg.values(), 'feature_names' : tfidf_features_names})
```

In [326]:

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

25 Negative features from TFIDF model

In [327]:

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

Out[327]:

	feature_names	feature_prob_estimates
4542	connecting	-14.168137
6610	fonts	-14.168137
6617	footballs	-14.168137
14120	worker	-14.168137
8519	linked	-14.168137
12374	standard	-14.168137
8520	linking	-14.168137
9309	nebraska	-14.168137
9304	nearly	-14.168137
1891	sun	-14.168137
6628	ford	-14.168137
12400	stash	-14.168137
1878	study	-14.168137
8537	listing	-14.168137
1869	strings	-14.168137

	feature_names	feature_prob_estimates
9750	overwhelmingly	-14.168137
6608	fondly	-14.168137
12349	stackable	-14.168137
1955	that	-14.168137
12297	sponsors	-14.168137
2054	urban	-14.168137
2052	upgrade	-14.168137
6573	flu	-14.168137
2032	tv	-14.168137
6583	flyers	-14.168137

In [328]:

```
tfidf_features_probs_pos = {}

for a in range(14244) :

    tfidf_features_probs_pos[a] = nb_tfidf.feature_log_prob_[1,a]
```

In [329]:

```
final_tfidf_features_pos = pd.DataFrame({'feature_prob_estimates' :
tfidf_features_probs_pos.values(), 'feature_names' : tfidf_features_names})
```

In [330]:

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

25 Positive features from TFIDF model

In [331]:

```
d.head(25)
```

Out[331]:

	feature_names	feature_prob_estimates
93	Grades_3-5	-15.895867
90	Grades_6-8	-15.895867
91	Grades_9-12	-15.895867
98	Dr.	-15.895867
97	Mr.	-15.895867
92	Grades_PreK-2	-15.895867
96	Ms.	-15.895867
95	Mrs.	-15.895867
94	nan	-15.895867
8286	ladder	-14.009989
1274	maximum	-13.923861
11467	royal	-13.882602
2283	350	-13.820593
8346	laughed	-13.762453

7966	intersection	-13.756510
10126	pilots	-13.726375
6388	features	-13.720271
7742	ineffective	-13.718434
3937	caucasians	-13.708868
1958	theatre	-13.679209
11468	rs	-13.661688
1535	pretty	-13.657959
4966	dash	-13.645893
10983	refills	-13.633327
12112	socialize	-13.628841

3. Conclusions

In [111]:

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

from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha:Hyper Parameter", "AUC"]

x.add_row(["BOW", "Naive Bayes", 0.5, 0.7])
x.add_row(["TFIDF", "Naive Bayes", 0.1, 0.67])

print(x)
```

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

In [112]:

```
y = PrettyTable()
y.field_names = ["Vectorizer", "Model", "K:Hyper Parameter", "AUC"]

y.add_row(["BOW", "KNN", 91, 0.63])
y.add_row(["TFIDF", "KNN", 85, 0.57])

print(y)
```

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

Summary

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

2. This can be observed by taking look at the difference in AUC measures for both the models

2. This can be observed by taking look at the difference in AUC measures for both the models. Clearly Naive Bayes is a better model.

3. Also, Naive Bayes takes very very less time to compute compared to the KNN model.