DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`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."
- __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 chart_studio.plotly import plot, iplot
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.sha
pe)
print('-'*50)
print("The attributes of data :", project_data.columns.values
)
```

```
Number of data points in train data (109248, 1
7)
--------------------------------------------------
----
The attributes of data : ['Unnamed: 0' 'id' 't
eacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_c
ategory'
 'project_subject_categories' 'project_subject
_subcategories'
 'project_title' 'project_essay_1' 'project_es
say_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.sh
ape)
```

```
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

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

1.2 preprocessing of `project_subject_categories`

In [5]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-pyth
on/
# https://stackoverflow.com/questions/23669024/how-to-strip-a
-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whit
espace-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 & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            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 ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc
", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the &
 value into
```

```
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:

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

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The " we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(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://stackoverf
low.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=l
ambda kv: kv[1]))
```

1.4 preprocessing of `teacher_prefix`

In [7]:

```
#NaN values in techer prefix will create a problem while enco
ding,so we replace NaN values with the mode of that particula
r column
#removing dot(.) since it is a special character
mode_of_teacher_prefix = project_data['teacher_prefix'].value
_counts().index[0]


project_data['teacher_prefix'] = project_data['teacher_prefix
'].fillna(mode_of_teacher_prefix)
```

In [8]:

```
prefixes = []

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

In [9]:

```
project_data.drop(['teacher_prefix'], axis = 1, inplace = Tru
e)
project_data["teacher_prefix"] = prefixes
print("After removing the special characters ,Column values:
 ")
np.unique(project_data["teacher_prefix"].values)
```

```
After removing the special characters ,Column
values:
```

Out[9]:

```
array(['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher'], dt
ype=object)
```

1.5 preprocessing of `school_state`

In [10]:

```
school_states = list(project_data['school_state'].values)
# remove special characters from list of strings python: http
s://stackoverflow.com/a/47301924/4084039

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

school_states_list = []
for i in school_states:
    temp = ""
    # consider we have text like this "Math & Science, Warmth
, Care & Hunger"
    for j in i.split(','): # it will split it in three parts
["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            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 ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc
", remove the trailing spaces
        temp = temp.replace('&','_')
    school_states_list.append(temp.strip())
```

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


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

school_states_dict = dict(my_counter)
sorted_school_states_dict = dict(sorted(school_states_dict.it
ems(), key=lambda kv: kv[1]))
```

1.6 preprocessing of `project_grade_category`

In [11]:

```
# We need to get rid of The spaces between the text and the h
yphens because they're special characters.
#Rmoving multiple characters from a string in Python
#https://stackoverflow.com/questions/3411771/multiple-charact
er-replace-with-python

project_grade_category = []

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

In [12]:

```
project_data.drop(['project_grade_category'], axis = 1, inpla
ce = True)
project_data["project_grade_category"] = project_grade_catego
ry
print("After removing the special characters ,Column values:
 ")
np.unique(project_data["project_grade_category"].values)
```

```
After removing the special characters ,Column
values:
```

Out[12]:

```
array(['Grades_3_5', 'Grades_6_8', 'Grades_9_1
2', 'Grades_PreK_2'], dtype=object)
```

1.3 Text preprocessing

In [13]:

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

In [14]:

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

Out[14]:

	Unnamed: 0	id	teacher_id
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc
1	140945	p258326	897464ce9ddc600bced1151f324dd63a

In [15]:

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

```
My students are English learners that are work
ing on English as their second or third langua
ges. We are a melting pot of refugees, immigra
nts, and native-born Americans bringing the gi
ft of language to our school. \r\n\r\n We have
 over 24 languages represented in our English
Learner program with students at every level o
f mastery.  We also have over 40 countries rep
resented with the families within our school.
 Each student brings a wealth of knowledge and
 experiences to us that open our eyes to new c
ultures, beliefs, and respect.\"The limits of
your language are the limits of your world.\"-
Ludwig Wittgenstein  Our English learner's hav
e a strong support system at home that begs fo
r more resources.  Many times our parents are
learning to read and speak English along side
of their children.  Sometimes this creates bar
riers for parents to be able to help their chi
ld learn phonetics, letter recognition, and ot
her reading skills.\r\n\r\nBy providing these
dvd's and players, students are able to contin
ue their mastery of the English language even
if no one at home is able to assist.  All fami
```

lies with students within the Level 1 proficie ncy status, will be a offered to be a part of this program. These educational videos will b e specially chosen by the English Learner Teac her and will be sent home regularly to watch. The videos are to help the child develop earl y reading skills.\r\n\r\nParents that do not h ave access to a dvd player will have the oppor tunity to check out a dvd player to use for th e year. The plan is to use these videos and e ducational dvd's for the years to come for oth er EL students.\r\nnannan

==========

The 51 fifth grade students that will cycle th rough my classroom this year all love learning , at least most of the time. At our school, 97 .3% of the students receive free or reduced pr ice lunch. Of the 560 students, 97.3% are mino rity students. \r\nThe school has a vibrant co mmunity that loves to get together and celebra te. Around Halloween there is a whole school p arade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a bi g festival with crafts made by the students, d ances, and games. At the end of the year the s chool hosts a carnival to celebrate the hard w ork put in during the school year, with a dunk tank being the most popular activity.My stude nts will use these five brightly colored Hokki stools in place of regular, stationary, 4-leg ged chairs. As I will only have a total of ten in the classroom and not enough for each stud ent to have an individual one, they will be us ed in a variety of ways. During independent re ading time they will be used as special chairs students will each use on occasion. I will ut

ilize them in place of chairs at my small grou
p tables during math and reading times. The re
st of the day they will be used by the student
s who need the highest amount of movement in t
heir life in order to stay focused on school.\
r\n\r\nWhenever asked what the classroom is mi
ssing, my students always say more Hokki Stool
s. They can't get their fill of the 5 stools w
e already have. When the students are sitting
in group with me on the Hokki Stools, they are
 always moving, but at the same time doing the
ir work. Anytime the students get to pick wher
e they can sit, the Hokki Stools are the first
 to be taken. There are always students who he
ad over to the kidney table to get one of the
stools who are disappointed as there are not e
nough of them. \r\n\r\nWe ask a lot of student
s to sit for 7 hours a day. The Hokki stools w
ill be a compromise that allow my students to
do desk work and move at the same time. These
stools will help students to meet their 60 min
utes a day of movement by allowing them to act
ivate their core muscles for balance while the
y sit. For many of my students, these chairs w
ill take away the barrier that exists in schoo
ls for a child who can't sit still.nannan
==========
====
How do you remember your days of school? Was i
t in a sterile environment with plain walls, r
ows of desks, and a teacher in front of the ro
om? A typical day in our room is nothing like
that. I work hard to create a warm inviting th
emed room for my students look forward to comi
ng to each day.\r\n\r\nMy class is made up of
28 wonderfully unique boys and girls of mixed
races in Arkansas.\r\nThey attend a Title I sc

hool, which means there is a high enough perce
ntage of free and reduced-price lunch to quali
fy. Our school is an \"open classroom\" concep
t, which is very unique as there are no walls
separating the classrooms. These 9 and 10 year
-old students are very eager learners; they ar
e like sponges, absorbing all the information
and experiences and keep on wanting more.With
these resources such as the comfy red throw pi
llows and the whimsical nautical hanging decor
 and the blue fish nets, I will be able to hel
p create the mood in our classroom setting to
be one of a themed nautical environment. Creat
ing a classroom environment is very important
in the success in each and every child's educa
tion. The nautical photo props will be used wi
th each child as they step foot into our class
room for the first time on Meet the Teacher ev
ening. I'll take pictures of each child with t
hem, have them developed, and then hung in our
 classroom ready for their first day of 4th gr
ade. This kind gesture will set the tone befo
re even the first day of school! The nautical
thank you cards will be used throughout the ye
ar by the students as they create thank you ca
rds to their team groups.\r\n\r\nYour generous
 donations will help me to help make our class
room a fun, inviting, learning environment fro
m day one.\r\n\r\nIt costs lost of money out o
f my own pocket on resources to get our classr
oom ready. Please consider helping with this p
roject to make our new school year a very succ
essful one. Thank you!nannan
==========
====
My kindergarten students have varied disabilit
ies ranging from speech and language delays, c

ognitive delays, gross/fine motor delays, to a utism. They are eager beavers and always striv e to work their hardest working past their lim itations. \r\n\r\nThe materials we have are th e ones I seek out for my students. I teach in a Title I school where most of the students re ceive free or reduced price lunch. Despite th eir disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had an ts in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chai rs are the answer and I love then because they develop their core, which enhances gross moto r and in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't wa nt to sit and do worksheets. They want to lear n to count by jumping and playing. Physical en gagement is the key to our success. The number toss and color and shape mats can make that h appen. My students will forget they are doing work and just have the fun a 6 year old deserv es.nannan

==========

The mediocre teacher tells. The good teacher e xplains. The superior teacher demonstrates. Th e great teacher inspires. -William A. Ward\r\n \r\nMy school has 803 students which is makeup is 97.6% African-American, making up the larg est segment of the student body. A typical sch ool in Dallas is made up of 23.2% African-Amer ican students. Most of the students are on fre e or reduced lunch. We aren't receiving doctor s, lawyers, or engineers children from rich ba

```
ckgrounds or neighborhoods. As an educator I a
m inspiring minds of young children and we foc
us not only on academics but one smart, effect
ive, efficient, and disciplined students with
good character.In our classroom we can utilize
 the Bluetooth for swift transitions during cl
ass. I use a speaker which doesn't amplify the
 sound enough to receive the message. Due to t
he volume of my speaker my students can't hear
 videos or books clearly and it isn't making t
he lessons as meaningful. But with the bluetoo
th speaker my students will be able to hear an
d I can stop, pause and replay it at any time.
\r\nThe cart will allow me to have more room f
or storage of things that are needed for the d
ay and has an extra part to it I can use.  The
 table top chart has all of the letter, words
and pictures for students to learn about diffe
rent letters and it is more accessible.nannan
==============================================
====
```

In [16]:

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

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

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
```

```
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [17]:

```
sent = decontracted(project_data['essay'].values[30000])
print(sent)
print("="*50)
```

```
My students are a highly mobile population mad
e up of low-income students and families from
a nearby military base. They thrive with a bal
anced approach of high expectations and positi
ve reinforcement, especially as quite a few of
 my students will attend several schools in th
e elementary years alone. Many of my students
struggle in the classroom, but have success in
 specialized classes like music and art. I str
ive to make my art room a space that is creati
ve yet disciplined, structured yet welcoming -
 a safe space for everyone.My mission as an ar
t teacher is to show my students that art, lik
e all things in life, is a process. It takes c
reativity, discipline and perseverance to make
 a product that one can take pride in. These l
ife skills will benefit my students far beyond
 art.\r\n\r\nMy students need weaving tools li
ke looms, canvas circles, and needles to build
 patience, fine motor skills, discipline and c
ooperation.\r\nWeaving is a universal craft th
at give me the opportunity to teach about many
 cultures as well as make fun stuff with my st
udents. I hope to create several cooperative t
apestries with the large looms in hopes that i
t will \"weave us together\" as a community!na
```

```
nnan
==========================================
====
```

In [18]:

```
# \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', ' ')
sent = sent.replace('nan',' ')
print(sent)
```

```
My students are a highly mobile population mad
e up of low-income students and families from
a nearby military base. They thrive with a bal
anced approach of high expectations and positi
ve reinforcement, especially as quite a few of
 my students will attend several schools in th
e elementary years alone. Many of my students
struggle in the classroom, but have success in
 specialized classes like music and art. I str
ive to make my art room a space that is creati
ve yet disciplined, structured yet welcoming -
 a safe space for everyone.My mission as an ar
t teacher is to show my students that art, lik
e all things in life, is a process. It takes c
reativity, discipline and perseverance to make
 a product that one can take pride in. These l
ife skills will benefit my students far beyond
 art.    My students need weaving tools like l
ooms, canvas circles, and needles to build pat
ience, fine motor skills, discipline and coope
ration.  Weaving is a universal craft that giv
e me the opportunity to teach about many cultu
res as well as make fun stuff with my students
. I hope to create several cooperative tapestr
```

```
ies with the large looms in hopes that it will
  weave us together  as a community!
```

In [19]:

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

```
My students are a highly mobile population mad
e up of low income students and families from
a nearby military base They thrive with a bala
nced approach of high expectations and positiv
e reinforcement especially as quite a few of m
y students will attend several schools in the
elementary years alone Many of my students str
uggle in the classroom but have success in spe
cialized classes like music and art I strive t
o make my art room a space that is creative ye
t disciplined structured yet welcoming a safe
space for everyone My mission as an art teache
r is to show my students that art like all thi
ngs in life is a process It takes creativity d
iscipline and perseverance to make a product t
hat one can take pride in These life skills wi
ll benefit my students far beyond art My stude
nts need weaving tools like looms canvas circl
es and needles to build patience fine motor sk
ills discipline and cooperation Weaving is a u
niversal craft that give me the opportunity to
 teach about many cultures as well as make fun
 stuff with my students I hope to create sever
al cooperative tapestries with the large looms
 in hopes that it will weave us together as a
community
```

In [20]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', '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', 'each', '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', "doesn'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"]
```

In [21]:

```
# Combining all the above statements
```

```
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')

    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)

    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopw
ords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:45<00:00, 2
407.65it/s]
```

In [22]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[22]:

```
'my kindergarten students varied disabilities
ranging speech language delays cognitive delay
s gross fine motor delays autism they eager be
avers always strive work hardest working past
limitations the materials ones i seek students
 i teach title i school students receive free
reduced price lunch despite disabilities limit
ations students love coming school come eager
learn explore have ever felt like ants pants n
eeded groove move meeting this kids feel time
the want able move learn say wobble chairs ans
wer i love develop core enhances gross motor t
urn fine motor skills they also want learn gam
```

```
es kids not want sit worksheets they want lear
n count jumping playing physical engagement ke
y success the number toss color shape mats mak
e happen my students forget work fun 6 year ol
d deserves nannan'
```

In [23]:

```
#creating a new column with the preprocessed essays and repla
cing it with the original columns
project_data['preprocessed_essays'] = preprocessed_essays
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
```

1.4 Preprocessing of $project_tit \leq$

In [24]:

```
# similarly you can preprocess the titles also

# similarly you can preprocess the titles also
preprocessed_titles = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].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 not in stopw
ords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:01<00:00, 5
7127.26it/s]
```

In [25]:

```
#creating a new column with the preprocessed titles,useful fo
r analysis
project_data['preprocessed_titles'] = preprocessed_titles
```

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

In [26]:

```
from sklearn.model_selection import train_test_split
#How to split whole dataset into Train,CV and test
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split
project_data_train, project_data_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'])
project_data_train, project_data_cv, y_train, y_cv = train_test_split(project_data_train, y_train, test_size=0.33, stratify=y_train)
```

In [27]:

```
print("Split ratio")
print('-'*50)
print('Train dataset:',len(project_data_train)/len(project_data)*100,'%\n','size:',len(project_data_train))
print('Cross validation dataset:',len(project_data_cv)/len(project_data)*100,'%\n','size:',len(project_data_cv))
print('Test dataset:',len(project_data_test)/len(project_data)*100,'%\n','size:',len(project_data_test))
```

```
Split ratio
--------------------------------------------------
----
Train dataset: 44.889608963093146 %
 size: 49041
```

```
Cross validation dataset: 22.110244581136495 %
 size: 24155
Test dataset: 33.000146455770356 %
 size: 36052
```

In [28]:

```
#Features
project_data_train.drop(['project_is_approved'], axis=1, inpl
ace=True)
project_data_cv.drop(['project_is_approved'], axis=1, inplace
=True)
project_data_test.drop(['project_is_approved'], axis=1, inpla
ce=True)
```

1.5 Preparing data for models

In [29]:

```
project_data.columns
```

Out[29]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'proj
ect_submitted_datetime',
       'project_title', 'project_resource_summ
ary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'clean_categories', 'clean_subcategorie
s', 'teacher_prefix',
       'school_state', 'project_grade_category
', 'essay',
       'preprocessed_essays', 'preprocessed_ti
tles'],
      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 (optional)

      - quantity : numerical (optional)
      - teacher_number_of_previously_posted_projects : nu
   merical
```

- price : numerical

2.2 Make Data Model Ready: encoding numerical, categorical features

1.5.1 Vectorizing Categorical data

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

In [30]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_cat.fit(project_data_train['clean_categories'].values) #fitting has to be on Train data

train_categories_one_hot = vectorizer_cat.transform(project_data_train['clean_categories'].values)
cv_categories_one_hot = vectorizer_cat.transform(project_data_cv['clean_categories'].values)
test_categories_one_hot = vectorizer_cat.transform(project_data_test['clean_categories'].values)


print(vectorizer_cat.get_feature_names())
print("Shape of training data matrix after one hot encoding "
,train_categories_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_categories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
```

```
t_categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'M
usic_Arts', 'AppliedLearning', 'SpecialNeeds',
 'Health_Sports', 'Math_Science', 'Literacy_La
nguage']
Shape of training data matrix after one hot en
coding  (49041, 9)
Shape of cross validation data matrix after on
e hot encoding  (24155, 9)
Shape of test data matrix after one hot encodi
ng  (36052, 9)
```

In [31]:

```
# we use count vectorizer to convert the values into one
vectorizer_subcat = CountVectorizer(vocabulary=list(sorted_su
b_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_subcat.fit(project_data_train['clean_subcategories'
].values)


train_subcategories_one_hot = vectorizer_subcat.transform(pro
ject_data_train['clean_subcategories'].values)
cv_subcategories_one_hot = vectorizer_subcat.transform(projec
t_data_cv['clean_subcategories'].values)
test_subcategories_one_hot = vectorizer_subcat.transform(proj
ect_data_test['clean_subcategories'].values)


print(vectorizer_subcat.get_feature_names())


print("Shape of train data matrix after one hot encoding ",tr
ain_subcategories_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_subcategories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
```

```
t_subcategories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLi
teracy', 'ParentInvolvement', 'Extracurricular
', 'Civics_Government', 'ForeignLanguages', 'N
utritionEducation', 'Warmth', 'Care_Hunger', '
SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_Car
eerPrep', 'Music', 'History_Geography', 'Healt
h_LifeScience', 'EarlyDevelopment', 'ESL', 'Gy
m_Fitness', 'EnvironmentalScience', 'VisualArt
s', 'Health_Wellness', 'AppliedSciences', 'Spe
cialNeeds', 'Literature_Writing', 'Mathematics
', 'Literacy']
Shape of train data matrix after one hot encod
ing  (49041, 30)
Shape of cross validation data matrix after on
e hot encoding  (24155, 30)
Shape of test data matrix after one hot encodi
ng  (36052, 30)
```

In [32]:

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

vectorizer_school_state = CountVectorizer()
vectorizer_school_state.fit(project_data_train['school_state']
.values)



print(vectorizer_school_state.get_feature_names())

train_school_state_category_one_hot = vectorizer_school_state
.transform(project_data_train['school_state'].values)
```

```
cv_school_state_category_one_hot = vectorizer_school_state.tr
ansform(project_data_cv['school_state'].values)
test_school_state_category_one_hot = vectorizer_school_state.
transform(project_data_test['school_state'].values)


print("Shape of train data matrix after one hot encoding ",tr
ain_school_state_category_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_school_state_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_school_state_category_one_hot.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc
', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', '
in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi',
 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh
', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', '
pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va',
 'vt', 'wa', 'wi', 'wv', 'wy']
Shape of train data matrix after one hot encod
ing  (49041, 51)
Shape of cross validation data matrix after on
e hot encoding  (24155, 51)
Shape of test data matrix after one hot encodi
ng  (36052, 51)
```

In [33]:

```
#This step is to intialize a vectorizer with vocab from train
 data
my_counter = Counter()
for project_grade in project_data_train['project_grade_catego
ry'].values:
    my_counter.update(project_grade.split())
```

In [34]:

```

```

```
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 [35]:

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

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_pro
ject_grade_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_grade.fit(project_data_train['project_grade_catego
ry'].values)



print(vectorizer_grade.get_feature_names())

train_project_grade_category_one_hot = vectorizer_grade.trans
form(project_data_train['project_grade_category'].values)
cv_project_grade_category_one_hot = vectorizer_grade.transfor
m(project_data_cv['project_grade_category'].values)
test_project_grade_category_one_hot = vectorizer_grade.transf
orm(project_data_test['project_grade_category'].values)


print("Shape of train data matrix after one hot encoding ",tr
ain_project_grade_category_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_project_grade_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_project_grade_category_one_hot.shape)
```

```
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'G
rades_PreK_2']
Shape of train data matrix after one hot encod
ing  (49041, 4)
Shape of cross validation data matrix after on
```

```
e hot encoding  (24155, 4)
Shape of test data matrix after one hot encodi
ng  (36052, 4)
```

In [36]:

```
#https://stackoverflow.com/questions/39303912/tfidfvectorizer
-in-scikit-learn-valueerror-np-nan-is-an-invalid-document
#ValueError: np.nan is an invalid document, expected byte or
unicode string.
vectorizer_prefix = CountVectorizer()
vectorizer_prefix.fit(project_data_train['teacher_prefix'].va
lues.astype("U"))


print(vectorizer_prefix.get_feature_names())


train_teacher_prefix_categories_one_hot = vectorizer_prefix.t
ransform(project_data_train['teacher_prefix'].values.astype("
U"))
cv_teacher_prefix_categories_one_hot = vectorizer_prefix.tran
sform(project_data_cv['teacher_prefix'].values.astype("U"))
test_teacher_prefix_categories_one_hot = vectorizer_prefix.tr
ansform(project_data_test['teacher_prefix'].values.astype("U"
))

print("Shape of train data matrix after one hot encoding ",tr
ain_teacher_prefix_categories_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_teacher_prefix_categories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_teacher_prefix_categories_one_hot.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of train data matrix after one hot encod
ing  (49041, 5)
Shape of cross validation data matrix after on
```

```
e hot encoding  (24155, 5)
Shape of test data matrix after one hot encodi
ng  (36052, 5)
```

Make Data Model Ready: encoding essay, and project_title

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [37]:

```
# We are considering only the words which appeared in at leas
t 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(project_data_train['preprocessed_ess
ays'].values)  #Fitting has to be on Train data


train_essay_bow = vectorizer_bow_essay.transform(project_data
_train['essay'].values)
cv_essay_bow = vectorizer_bow_essay.transform(project_data_cv
['essay'].values)
test_essay_bow = vectorizer_bow_essay.transform(project_data_
test['essay'].values)


print("Shape of train data matrix after one hot encoding ",tr
ain_essay_bow.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_essay_bow.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_essay_bow.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 12130)
Shape of cross validation data matrix after on
e hot encoding  (24155, 12130)
Shape of test data matrix after one hot encodi
ng  (36052, 12130)
```

In [38]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit_transform(project_data_train['prepro
cessed_titles'].values)    #Fitting has to be on Train data


train_title_bow = vectorizer_bow_title.transform(project_data
_train['preprocessed_titles'].values)
cv_title_bow = vectorizer_bow_title.transform(project_data_cv
['preprocessed_titles'].values)
test_title_bow = vectorizer_bow_title.transform(project_data_
test['preprocessed_titles'].values)



print("Shape of train data matrix after one hot encoding ",tr
ain_title_bow.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_title_bow.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_title_bow.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 2093)
Shape of cross validation data matrix after on
e hot encoding  (24155, 2093)
Shape of test data matrix after one hot encodi
ng  (36052, 2093)
```

1.5.2.2 TFIDF vectorizer

In [39]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(project_data_train['preprocessed_essays'])      #Fitting has to be on Train data

train_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_train['preprocessed_essays'].values)
cv_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_cv['preprocessed_essays'].values)
test_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_test['preprocessed_essays'].values)

print("Shape of train data matrix after one hot encoding ",train_essay_tfidf.shape)
print("Shape of cross validation data matrix after one hot encoding ",cv_essay_tfidf.shape)
print("Shape of test data matrix after one hot encoding ",test_essay_tfidf.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 12130)
Shape of cross validation data matrix after on
e hot encoding  (24155, 12130)
Shape of test data matrix after one hot encodi
ng  (36052, 12130)
```

In [40]:

```
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(project_data_train['preprocessed_titles'])      #Fitting has to be on Train data
```

```
train_title_tfidf = vectorizer_tfidf_title.transform(project_
data_train['preprocessed_titles'].values)
cv_title_tfidf = vectorizer_tfidf_title.transform(project_dat
a_cv['preprocessed_titles'].values)
test_title_tfidf = vectorizer_tfidf_title.transform(project_d
ata_test['preprocessed_titles'].values)

print("Shape of train data matrix after one hot encoding ",tr
ain_title_tfidf.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_title_tfidf.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_title_tfidf.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 2093)
Shape of cross validation data matrix after on
e hot encoding  (24155, 2093)
Shape of test data matrix after one hot encodi
ng  (36052, 2093)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [41]:

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

In [42]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_avg_w2v_essays.append(vector)

print(len(train_avg_w2v_essays))
print(len(train_avg_w2v_essays[0]))
```

```
100%|██████████| 49041/49041 [00:08<00:00, 5836.82it/s]
```

```
49041
300
```

In [43]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essays']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
```

```
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    cv_avg_w2v_essays.append(vector)

print(len(cv_avg_w2v_essays))
print(len(cv_avg_w2v_essays[0]))
```

```
100%|██████████| 24155/24155 [00:04<00:00, 582
3.72it/s]
```

```
24155
300
```

In [44]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_essays = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essays']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
```

```
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_avg_w2v_essays.append(vector)

print(len(test_avg_w2v_essays))
print(len(test_avg_w2v_essays[0]))
```

```
100%|██████████| 36052/36052 [00:06<00:00, 584
1.95it/s]
```

```
36052
300
```

In [45]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_titles = []; # the avg-w2v for each sentence/re
view is stored in this list
for sentence in tqdm(project_data_train['preprocessed_titles'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_avg_w2v_titles.append(vector)

print(len(train_avg_w2v_titles))
print(len(train_avg_w2v_titles[0]))
```

```
100%|██████████| 49041/49041 [00:00<00:00, 100
132.05it/s]
```

```
49041
300
```

In [46]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_avg_w2v_titles = []; # the avg-w2v for each sentence/revie
w is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_titles']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    cv_avg_w2v_titles.append(vector)

print(len(cv_avg_w2v_titles))
print(len(cv_avg_w2v_titles[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 994
29.13it/s]
```

```
24155
300
```

In [47]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_avg_w2v_titles.append(vector)

print(len(test_avg_w2v_titles))
print(len(test_avg_w2v_titles[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 99531.84it/s]
```

```
36052
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [48]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_essays'].val
```

```
ues)
# we are converting a dictionary with word as a key, and the
idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [49]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_essays = []; # the avg-w2v for each sentence/
review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_essays'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_essays.append(vector)

print(len(train_tfidf_w2v_essays))
```

```
print(len(train_tfidf_w2v_essays[0]))
```

```
100%|██████████| 49041/49041 [01:04<00:00, 764
.98it/s]
```

```
49041
300
```

In [50]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_tfidf_w2v_essays = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essays']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_tfidf_w2v_essays.append(vector)
```

```
print(len(cv_tfidf_w2v_essays))
print(len(cv_tfidf_w2v_essays[0]))
```

```
100%|██████████| 24155/24155 [00:31<00:00, 770
.68it/s]
```

```
24155
300
```

In [51]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_essays = []; # the avg-w2v for each sentence/r
eview is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essays']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_essays.append(vector)
```

```
print(len(test_tfidf_w2v_essays))
print(len(test_tfidf_w2v_essays[0]))
```

```
100%|██████████| 36052/36052 [00:46<00:00, 767
.27it/s]
```

```
36052
300
```

In [52]:

```
# Similarly you can vectorize for title also
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_titles'])
# we are converting a dictionary with word as a key, and the
idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [53]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_titles = []; # the avg-w2v for each sentence/
review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_titles'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
ord
```

```
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_titles.append(vector)

print(len(train_tfidf_w2v_titles))
print(len(train_tfidf_w2v_titles[0]))
```

```
100%|██████████| 49041/49041 [00:01<00:00, 44641.31it/s]
```

```
49041
300
```

In [54]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_tfidf_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each w
```

```
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_tfidf_w2v_titles.append(vector)

print(len(cv_tfidf_w2v_titles))
print(len(cv_tfidf_w2v_titles[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 446
16.72it/s]
```

```
24155
300
```

In [55]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_titles = []; # the avg-w2v for each sentence/r
eview is stored in this list
for sentence in tqdm(project_data_test['preprocessed_titles']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he sentence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if (word in glove_words) and (word in tfidf_words):
```

```
            vec = model[word] # getting the vector for each w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_titles.append(vector)

print(len(test_tfidf_w2v_titles))
print(len(test_tfidf_w2v_titles[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 452
11.10it/s]
```

```
36052
300
```

1.5.3 Vectorizing Numerical features

In [56]:

```
price_data = resource_data.groupby('id').agg({'price':'sum',
'quantity':'sum'}).reset_index()
```

In [57]:

```
project_data_train = pd.merge(project_data_train, price_data,
 on='id', how='left')
project_data_cv = pd.merge(project_data_cv, price_data, on='i
d', how='left')
```

```
project_data_test = pd.merge(project_data_test, price_data, o
n='id', how='left')
```

In [58]:

```
from sklearn.preprocessing import Normalizer
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer = Normalizer()
normalizer.fit(project_data_train['price'].values.reshape(-1,1
))


price_normalized_train = normalizer.transform(project_data_tr
ain['price'].values.reshape(-1, 1))
price_normalized_cv = normalizer.transform(project_data_cv['p
rice'].values.reshape(-1, 1))
price_normalized_test = normalizer.transform(project_data_tes
t['price'].values.reshape(-1, 1))


print('After normalization')
print(price_normalized_train.shape)
print(price_normalized_cv.shape)
print(price_normalized_test.shape)
```

```
After normalization
(49041, 1)
(24155, 1)
(36052, 1)
```

In [59]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))


# Now standardize the data with above maen and variance.
previously_posted_projects_normalized_train = normalizer.transform(project_data_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
previously_posted_projects_normalized_cv = normalizer.transform(project_data_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
previously_posted_projects_normalized_test = normalizer.transform(project_data_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))


print('After normalization')
print(previously_posted_projects_normalized_train.shape)
print(previously_posted_projects_normalized_cv.shape)
print(previously_posted_projects_normalized_test.shape)
```

```
After normalization
(49041, 1)
(24155, 1)
(36052, 1)
```

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**

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

2. **The hyper paramter tuning(find best Alpha)**

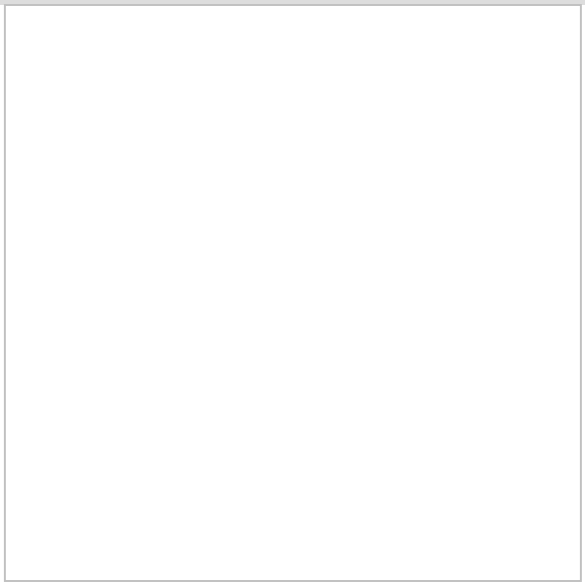
    - Find the best hyper parameter which will give the maximum AUC value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch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

3. **Feature importance**

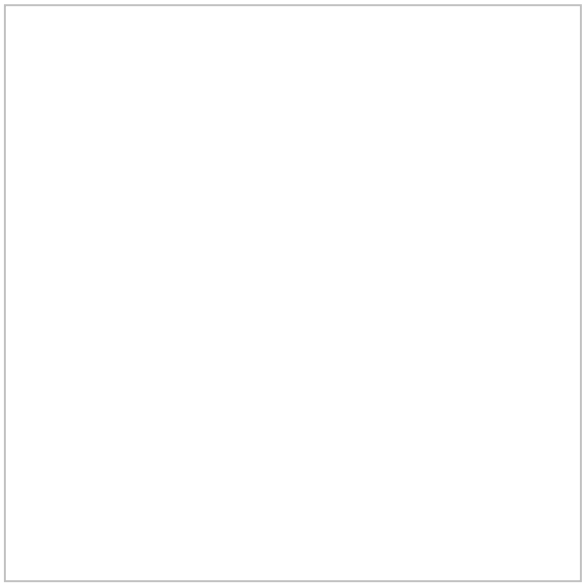
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB and print their corresponding feature names

4. **Representation of results**

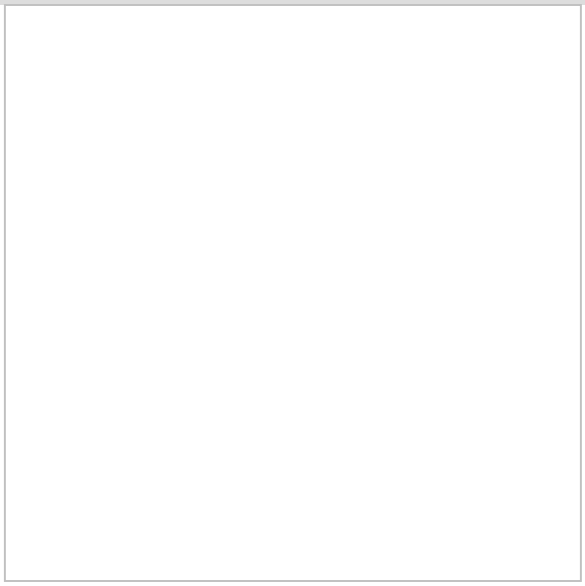
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.



- 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.

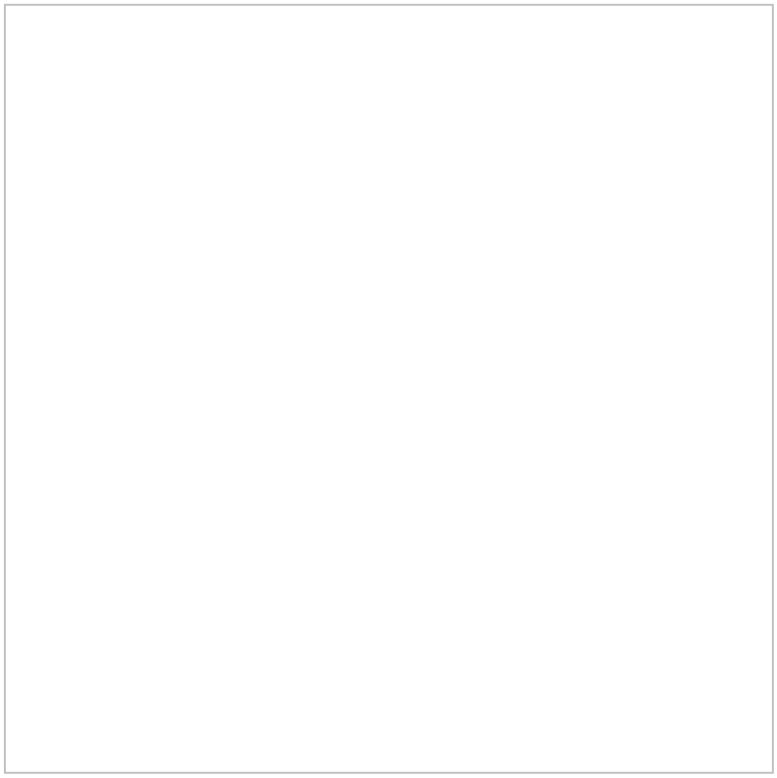


- Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.



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

Applying Naive Bayes on BOW, SET 1

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

In [60]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_bow, train_title_bow, train_school_state_category_one_hot,train_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_train, train_project_grade_category_one_hot, price_normalized_train)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_essay_bow, cv_title_bow, cv_school_state_category_one_hot, cv_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_cv, cv_project_grade_category_one_hot, price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_bow, test_title_bow, test_school_state_category_one_hot, test_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_test, test_project_grade_category_one_hot, price_normalized_test)).tocsr()

print(X_train.shape)
print(X_cv.shape)
```

```
print(X_test.shape)
```

```
(49041, 14324)
(24155, 14324)
(36052, 14324)
```

In [61]:

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

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

    return y_data_pred
```

In [62]:

```
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 = []
```

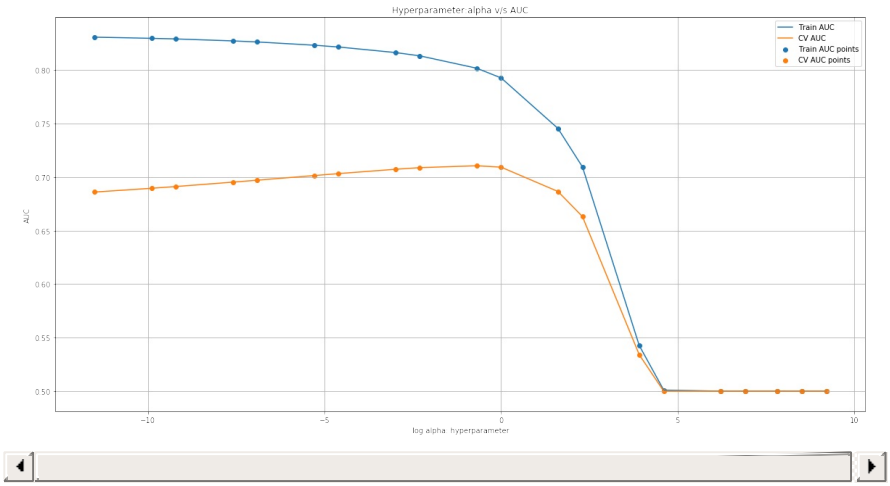
```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.0
1, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000,
10000]

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

    y_train_pred = batch_predict(nb, X_train)
    y_cv_pred = batch_predict(nb, X_cv)


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

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

```
100%|██████████| 20/20 [00:02<00:00,  7.58it/s
]
100%|██████████| 20/20 [00:00<00:00, 88115.63i
t/s]
```

In [63]:

```
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("Hyperparameter:alpha v/s AUC")
plt.grid()
plt.show()
```



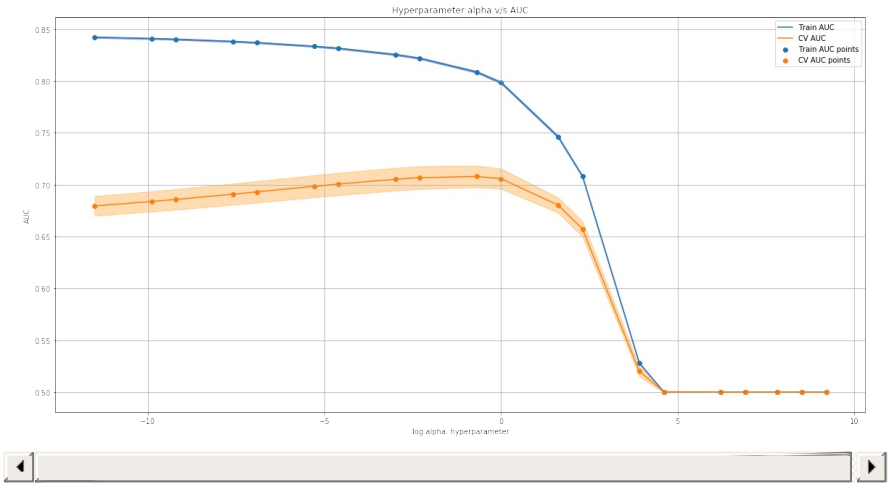
In [64]:

```
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB()

parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.00
1, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000,
 2500, 5000, 10000]}

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

clf.fit(X_train, y_train)

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

In [65]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.0
1, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000,
10000]
log_alphas =[]

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

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

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(log_alphas,train_auc - train_auc_std,t
rain_auc + train_auc_std,alpha=0.3,color='darkblue')

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

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


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

```
100%|██████████| 20/20 [00:00<00:00, 97769.32i
t/s]
```



In [66]:

```
#https://datascience.stackexchange.com/questions/21877/how-to-use-the-output-of-gridsearch
#choosing the best hyperparameter
clf.best_params_
```

Out[66]:

```
{'alpha': 0.5}
```

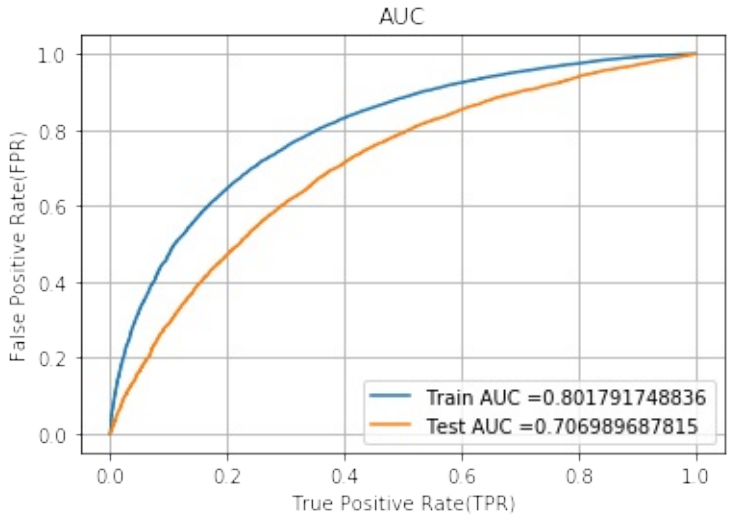
In [67]:

```
best_alpha1=clf.best_params_['alpha']
```

In [68]:

```
# 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_alpha1)

nb_bow.fit(X_train, 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_train)
y_test_pred = batch_predict(nb_bow, X_test)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_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(tr
ain_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()
```



In [69]:

```
# we are writing our own function for predict, with defined t
hresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshold, fpr, tpr):
```

```
    t = threshold[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshold):
    predictions = []
    for i in proba:
        if i>=threshold:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [70]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.53231399340
2 for threshold 0.861
Train confusion matrix
[[ 5499  1927]
 [11700 29915]]
```

In [71]:

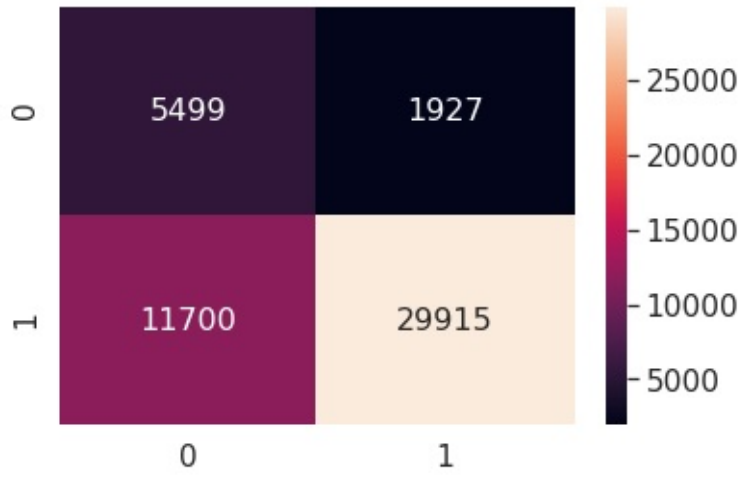
```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix
```

```
print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[71]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32b13e3d68>
```



In [72]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 3371  2088]
 [ 9263 21330]]
```

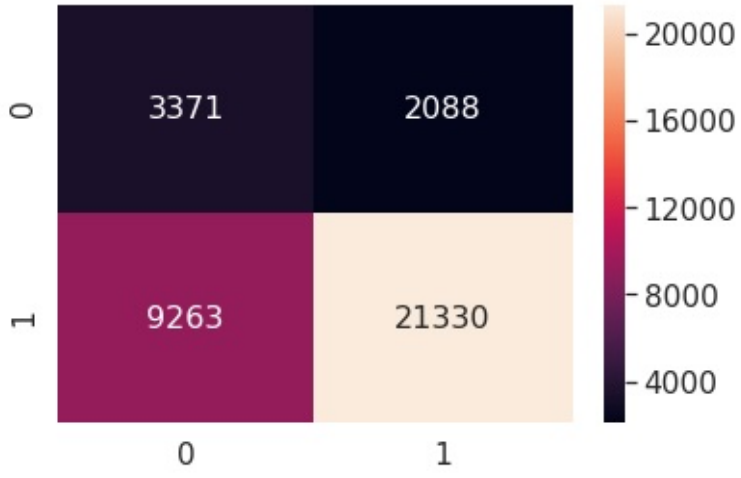
In [73]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32b1384080>
```



Applying Naive Bayes on TFIDF, SET 2

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

In [74]:

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

```
X_train = hstack((train_categories_one_hot, train_subcategori
es_one_hot, train_essay_tfidf, train_title_tfidf, train_schoo
l_state_category_one_hot,train_teacher_prefix_categories_one_
hot, previously_posted_projects_normalized_train, train_proje
ct_grade_category_one_hot, price_normalized_train)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_ho
t, cv_essay_tfidf, cv_title_tfidf, cv_school_state_category_o
ne_hot, cv_teacher_prefix_categories_one_hot, previously_post
ed_projects_normalized_cv, cv_project_grade_category_one_hot,
 price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_
one_hot, test_essay_tfidf, test_title_tfidf, test_school_stat
e_category_one_hot, test_teacher_prefix_categories_one_hot, p
reviously_posted_projects_normalized_test, test_project_grade
_category_one_hot, price_normalized_test)).tocsr()
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(49041, 14324)
(24155, 14324)
(36052, 14324)
```

In [75]:

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

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, 0.0
```

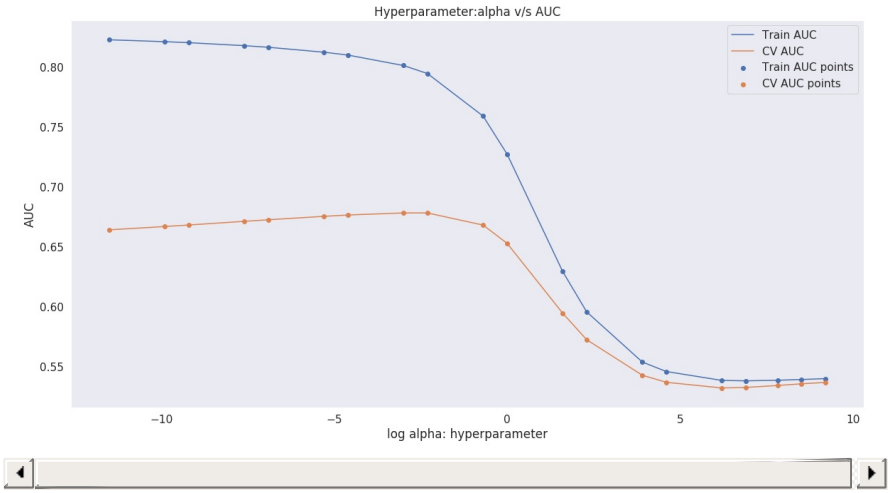
```
1, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000,
10000]

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

    y_train_pred = batch_predict(nb, X_train)
    y_cv_pred = batch_predict(nb, X_cv)


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

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

```
100%|██████████| 20/20 [00:02<00:00,  8.32it/s
]
100%|██████████| 20/20 [00:00<00:00, 91980.35i
t/s]
```

In [76]:

```
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("Hyperparameter:alpha v/s AUC")
plt.grid()
plt.show()
```



In [77]:

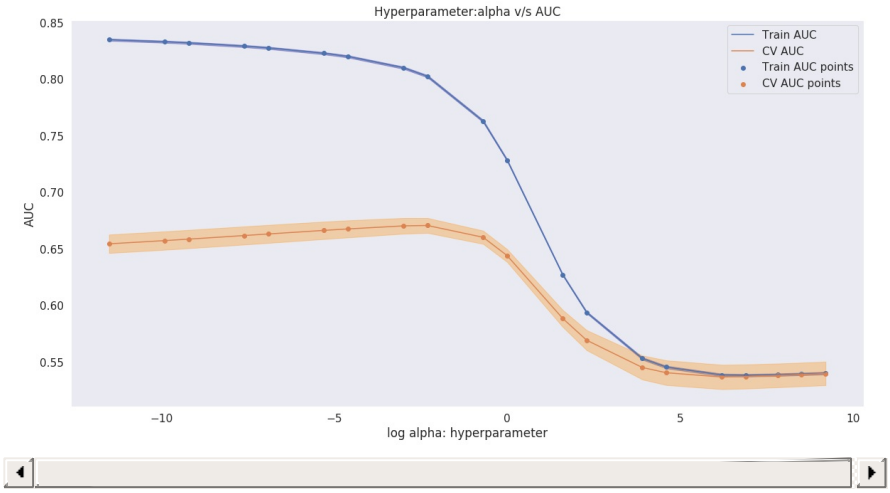
```
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB()

parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.00
1, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000,
 2500, 5000, 10000]}

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

clf.fit(X_train, y_train)

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

In [78]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.0
1, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000,
```

```
10000]
log_alphas =[]

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

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

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

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

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


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

```
100%|██████████| 20/20 [00:00<00:00, 153076.79 it/s]
```



In [79]:

```
#https://datascience.stackexchange.com/questions/21877/how-to-use-the-output-of-gridsearch
#choosing the best hyperparameter
clf.best_params_
```

Out[79]:

```
{'alpha': 0.1}
```

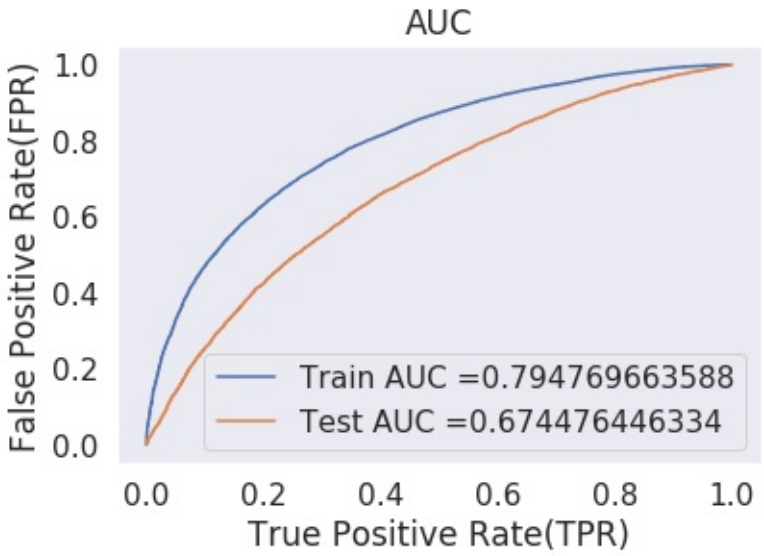
In [80]:

```
best_alpha2 = clf.best_params_['alpha']
```

In [81]:

```
# 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_alpha2)

nb_bow.fit(X_train, 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_train)
y_test_pred = batch_predict(nb_bow, X_test)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_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(tr
ain_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()
```



In [82]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
```

```
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.51945110455
7 for threshold 0.853
Train confusion matrix
[[ 5481  1945]
 [12327 29288]]
```

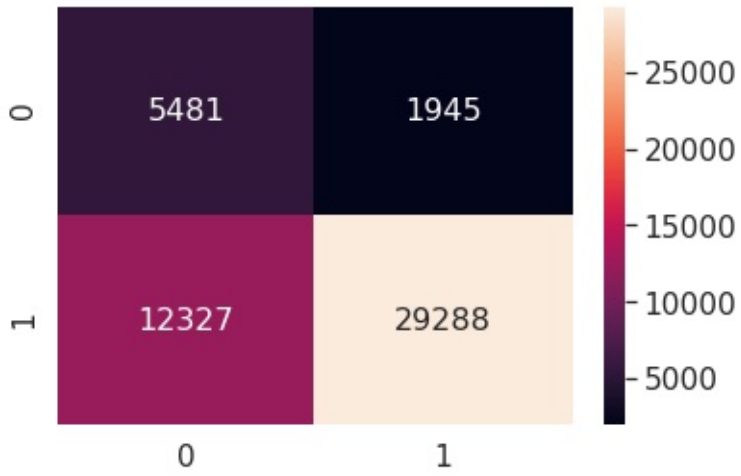
In [83]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[83]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
32b1639e10>
```



In [84]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 3166  2293]
 [ 9958 20635]]
```

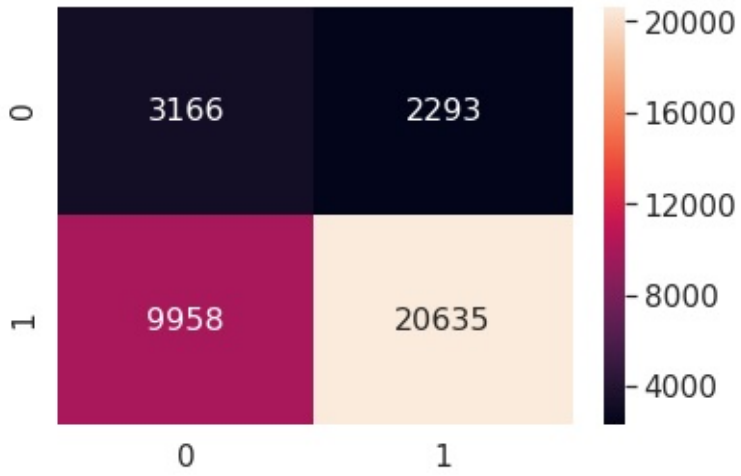
In [85]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[85]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f32b152c5f8>
```


3166
2293
9958
20635
0
1
0
1
20000
16000
12000
8000
4000

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

In [86]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_bow, train_title_bow, train_school_state_category_one_hot,train_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_train, train_project_grade_category_one_hot, price_normalized_train)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_essay_bow, cv_title_bow, cv_school_state_category_one_hot, cv_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_cv, cv_project_grade_category_one_hot, price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_bow, test_title_bow, test_school_state_category_one_hot, test_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_test, test_project_grade_category_one_hot, price_normalized_test)).tocsr()

print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(49041, 14324)
(24155, 14324)
(36052, 14324)
```

In [87]:

```
nb_bow = MultinomialNB(alpha = 0.5)

nb_bow.fit(X_train, y_train)
```

Out[87]:

```
MultinomialNB(alpha=0.5, class_prior=None, fit
_prior=True)
```

Top 10 important features of negative class from SET 1(BOW)

In [90]:

```
neg_bow_feature_probs = {}

for a in range(14324) :

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

In [91]:

```
len(neg_bow_feature_probs.values())
```

Out[91]:

```
14324
```

In [92]:

```
neg_bow_feature_prob_values = [ v for v in neg_bow_feature_pr
obs.values() ]
```

In [93]:

```
len(neg_bow_feature_prob_values)
```

Out[93]:

```
14324
```

In [94]:

```
neg_bow_features_names = []
```

In [95]:

```
for feature in vectorizer_cat.get_feature_names() :
    neg_bow_features_names.append(feature)

for feature in vectorizer_subcat.get_feature_names() :
    neg_bow_features_names.append(feature)

for feature in vectorizer_school_state.get_feature_names() :
    neg_bow_features_names.append(feature)

for feature in vectorizer_grade.get_feature_names() :
    neg_bow_features_names.append(feature)

for feature in vectorizer_prefix.get_feature_names() :
    neg_bow_features_names.append(feature)
```

In [96]:

```
for feature in vectorizer_bow_title.get_feature_names() :
    neg_bow_features_names.append(feature)
```

In [97]:

```
for feature in vectorizer_bow_essay.get_feature_names() :
    neg_bow_features_names.append(feature)
```

In [98]:

```
neg_bow_features_names.append("price")
```

```
neg_bow_features_names.append("teacher_number_of_previously_posted_projects")
```

In [99]:

```
len(neg_bow_features_names)
```

Out[99]:

14324

In [100]:

```
neg_bow_features_df = pd.DataFrame({'feature_prob_estimates' : neg_bow_feature_prob_values, 'feature_names' : neg_bow_features_names})
```

In [101]:

```
final_neg_bow_features = neg_bow_features_df.sort_values(by = ['feature_prob_estimates'], ascending = True)
```

In [104]:

```
final_neg_bow_features.head(10)
```

Out[104]:

	feature_prob_estimates	feature_names
2592	-15.110422	adopt
6053	-15.110422	epson
6047	-15.110422	envy
1635	-15.110422	relationships
13361	-15.110422	transportation
1617	-15.110422	ready
9929	-15.110422	pasts
12344	-15.110422	sponsors

1610	-15.110422	reaching
1601	-15.110422	rain

Top 10 important features of positive class from SET 1(BOW)

In [105]:

```
pos_bow_feature_probs = {}

for a in range(14324) :

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

In [106]:

```
len(pos_bow_feature_probs.values())
```

Out[106]:

14324

In [107]:

```
pos_bow_feature_prob_values = [ v for v in pos_bow_feature_pr
obs.values() ]
```

In [108]:

```
pos_bow_features_names = []
```

In [109]:

```
for feature in vectorizer_cat.get_feature_names() :
    pos_bow_features_names.append(feature)

for feature in vectorizer_subcat.get_feature_names() :
    pos_bow_features_names.append(feature)
```

```
for feature in vectorizer_school_state.get_feature_names() :
    pos_bow_features_names.append(feature)

for feature in vectorizer_grade.get_feature_names() :
    pos_bow_features_names.append(feature)

for feature in vectorizer_prefix.get_feature_names() :
    pos_bow_features_names.append(feature)
```

In [110]:

```
for feature in vectorizer_bow_title.get_feature_names() :
    pos_bow_features_names.append(feature)
```

In [111]:

```
for feature in vectorizer_bow_essay.get_feature_names() :
    pos_bow_features_names.append(feature)
```

In [112]:

```
pos_bow_features_names.append("price")

pos_bow_features_names.append("teacher_number_of_previously_posted_projects")
```

In [113]:

```
len(pos_bow_features_names)
```

Out[113]:

```
14324
```

In [114]:

```
pos_bow_features_df = pd.DataFrame({'feature_prob_estimates' : pos_bow_feature_prob_values, 'feature_names' : pos_bow_features_names})
```

In [115]:

```
final_pos_bow_features = pos_bow_features_df.sort_values(by =
 ['feature_prob_estimates'], ascending = True)
```

In [116]:

```
final_pos_bow_features.head(10)
```

Out[116]:

	feature_prob_estimates	feature_names
12011	-16.884522	simulations
815	-16.884522	film
12400	-14.486627	stadium
7451	-14.486627	honor
10969	-14.486627	recollect
3754	-14.486627	burgeoning
608	-14.486627	differentiate
13023	-14.486627	tether
5969	-14.486627	england
7928	-14.486627	instructors

Top 10 important features of negative class from SET 2(TFIDF)

In [117]:

```
# merge two sparse matrices: https://stackoverflow.com/a/1971
0648/4084039

# with the same hstack function we are concatinating a sparse
```

```
 matrix and a dense matirx :)
X_train = hstack((train_categories_one_hot, train_subcategori
es_one_hot, train_essay_tfidf, train_title_tfidf, train_schoo
l_state_category_one_hot,train_teacher_prefix_categories_one_
hot, previously_posted_projects_normalized_train, train_proje
ct_grade_category_one_hot, price_normalized_train)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_ho
t, cv_essay_tfidf, cv_title_tfidf, cv_school_state_category_o
ne_hot, cv_teacher_prefix_categories_one_hot, previously_post
ed_projects_normalized_cv, cv_project_grade_category_one_hot,
 price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_
one_hot, test_essay_tfidf, test_title_tfidf, test_school_stat
e_category_one_hot, test_teacher_prefix_categories_one_hot, p
reviously_posted_projects_normalized_test, test_project_grade
_category_one_hot, price_normalized_test)).tocsr()

print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(49041, 14324)
(24155, 14324)
(36052, 14324)
```

In [118]:

```
nb_tfidf = MultinomialNB(alpha = 0.1)

nb_tfidf.fit(X_train, y_train)
```

Out[118]:

```
MultinomialNB(alpha=0.1, class_prior=None, fit
_prior=True)
```

In [119]:

```
neg_tfidf_feature_probs = {}
```

```
for a in range(14324) :

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

In [120]:

```
neg_tfidf_feature_prob_values = [ v for v in neg_tfidf_featur
e_probs.values() ]
```

In [122]:

```
neg_tfidf_features_names = []
```

In [123]:

```
for feature in vectorizer_cat.get_feature_names() :
    neg_tfidf_features_names.append(feature)

for feature in vectorizer_subcat.get_feature_names() :
    neg_tfidf_features_names.append(feature)

for feature in vectorizer_school_state.get_feature_names() :
    neg_tfidf_features_names.append(feature)

for feature in vectorizer_grade.get_feature_names() :
    neg_tfidf_features_names.append(feature)

for feature in vectorizer_prefix.get_feature_names() :
    neg_tfidf_features_names.append(feature)
```

In [124]:

```
for feature in vectorizer_tfidf_title.get_feature_names() :
    neg_tfidf_features_names.append(feature)

for feature in vectorizer_tfidf_essay.get_feature_names() :
    neg_tfidf_features_names.append(feature)
```

In [125]:

```
neg_tfidf_features_names.append('price')
neg_tfidf_features_names.append('teacher_number_of_previously
_posted_projects')
```

In [126]:

```
len(neg_tfidf_features_names)
```

Out[126]:

14324

In [127]:

```
neg_tfidf_features_df = pd.DataFrame({'feature_prob_estimates
' : neg_tfidf_feature_prob_values, 'feature_names' : neg_tfid
f_features_names})
```

In [128]:

```
final_neg_tfidf_features = neg_tfidf_features_df.sort_values(
by = ['feature_prob_estimates'], ascending = True)
```

In [129]:

```
final_neg_tfidf_features.head(10)
```

Out[129]:

	feature_prob_estimates	feature_names
2755	-14.125782	alongside
13184	-14.125782	tinkering
8622	-14.125782	location
13191	-14.125782	tired
1826	-14.125782	spot

1808	-14.125782	spaces
10732	-14.125782	puncher
8615	-14.125782	loaned
8652	-14.125782	looming
8654	-14.125782	loop

Top 10 important features of positive class from SET 2(TFIDF)

In [130]:

```
pos_tfidf_feature_probs = {}

for a in range(14324) :

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

In [131]:

```
pos_tfidf_feature_prob_values = [ v for v in pos_tfidf_featur
e_probs.values() ]
```

In [132]:

```
pos_tfidf_features_names = []
```

In [133]:

```
for feature in vectorizer_cat.get_feature_names() :
    pos_tfidf_features_names.append(feature)

for feature in vectorizer_subcat.get_feature_names() :
    pos_tfidf_features_names.append(feature)
```

```
for feature in vectorizer_school_state.get_feature_names() :
    pos_tfidf_features_names.append(feature)

for feature in vectorizer_grade.get_feature_names() :
    pos_tfidf_features_names.append(feature)

for feature in vectorizer_prefix.get_feature_names() :
    pos_tfidf_features_names.append(feature)
```

In [134]:

```
for feature in vectorizer_tfidf_title.get_feature_names() :
    pos_tfidf_features_names.append(feature)

for feature in vectorizer_tfidf_essay.get_feature_names() :
    pos_tfidf_features_names.append(feature)
```

In [135]:

```
pos_tfidf_features_names.append('price')
pos_tfidf_features_names.append('teacher_number_of_previously
_posted_projects')
```

In [136]:

```
len(pos_tfidf_features_names)
```

Out[136]:

```
14324
```

In [137]:

```
pos_tfidf_features_df = pd.DataFrame({'feature_prob_estimates
' : pos_tfidf_feature_prob_values, 'feature_names' : pos_tfid
f_features_names})
```

In [138]:

```
final_pos_tfidf_features = pos_tfidf_features_df.sort_values(
```

```
by = ['feature_prob_estimates'], ascending = True)
```

In [139]:

```
final_pos_tfidf_features.head(10)
```

Out[139]:

	feature_prob_estimates	feature_names
7928	-13.846857	instructors
1107	-13.740583	journalism
5969	-13.739623	england
9047	-13.739391	million
6104	-13.714257	estimate
608	-13.713605	differentiate
3064	-13.681641	assistive
3754	-13.673101	burgeoning
1964	-13.649922	the
5858	-13.644879	eliminating

3. Conclusions

In [1]:

```
# Please compare all your models using Prettytable library
# 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 Paramete
r", "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 Param
eter | AUC  |
+------------+-------------+------------------
-----+------+
|    BOW     | Naive Bayes |          0.5
     | 0.7  |
|   TFIDF    | Naive Bayes |          0.1
     | 0.67 |
+------------+-------------+------------------
-----+------+
```