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	
`project_id`	A unique identifier for the proposed project. **Examp
`project_title`	Title of the proje • `Art Will Make` • `Firs`
`project_grade_category`	Grade level of students for which the project is targeted. One enum • `Gra` • • • `(`
`project_subject_categories`	One or more (comma-separated) subject categories for the p following enumerate • `Applie` • `Ca` • `Heal` • `Histo` • `Literacy` • `Mat` • `Music` • `Spe` • • `Music` • `Literacy & Language, Mat`
`school_state`	State where school is located ([Two-letter U (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#F
`project_subject_subcategories`	One or more (comma-separated) subject subcategories • • `Literature & Writing, Socia`
`project_resource_summary`	An explanation of the resources needed for the proj • `My students need hands on literacy material sens`
`project_essay_1`	First ap
`project_essay_2`	Second ap
`project_essay_3`	Third ap
`project_essay_4`	Fourth ap
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `1`
`teacher_id`	A unique identifier for the teacher of the proposed pro `bdf8baa8fedef6bfeec7a`

Feature	
	Teacher's title. One of the following enum
teacher_prefix	• • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the

[*] 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

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 [85]:

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

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

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

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

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

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
from scipy.sparse import hstack
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Normalizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from prettytable import PrettyTable
```

1.1 Reading Data

In [17]:

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

Adding price attribute to project_data dataframe from resources using merge function

In [18]:

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

In [19]:

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

```
Number of data points in train data (109248, 19)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_pr
efix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_
3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approve
d'
 'price' 'quantity']
```

In [20]:

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

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

Out[20]:

	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`

Here we perform the following operations

- Clean the project_subject_categrories by converting Math & Science, Care & Hunger ==> Math_Science Care_Hunger and put them in the cat_list.
- Remove the actual column from the pandas dataframe and instead include another column called 'cleaned categories'.
- Then create a dictonary which holds the frequency of the unique subject categories, if a project falls under two different categories then this will will increase the count of each of those subject categories by one. After cleaning the data multiple categories for a project are separated by space, hence, it is easy to split them and then use Counter to create the dictionary with frequency of each category.
- Sort the dictionary based on the frequency.

In [21]:

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

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

project_data['clean_categories'] = cat_list

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`

- We process the subject sub-categories in a manner similar to the subject categories.
- Then, create a dictionary with key as the sub-category and the value as the frequency .
- Sort the dictionary based on the frerquency.

In [22]:

```
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
```

In [23]:

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

1.3 Text preprocessing

In [24]:

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

In [25]:

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

Out[25]:

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

In [26]:

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

In [27]:

```
# 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[30000])
print("="*50)
```

```
My students are English learners that are working on English as thei
r second or third languages. We are a melting pot of refugees, immig
rants, and native-born Americans bringing the gift of language to ou
r school. \r\n\r\n We have over 24 languages represented in our Engl
ish Learner program with students at every level of mastery.  We als
o have over 40 countries represented with the families within our sc
hool.  Each student brings a wealth of knowledge and experiences to 
us that open our eyes to new cultures, beliefs, and respect.\"The li
mits of your language are the limits of your world.\"-Ludwig Wittgen
stein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to 
read and speak English along side of their children.  Sometimes this
creates barriers for parents to be able to help their child learn ph
onetics, letter recognition, and other reading skills.\r\n\r\nBy pro
viding these dvd's and players, students are able to continue their 
mastery of the English language even if no one at home is able to as
sist.  All families with students within the Level 1 proficiency sta
tus, will be a offered to be a part of this program.  These educatio
nal videos will be specially chosen by the English Learner Teacher a
nd will be sent home regularly to watch.  The videos are to help the
child develop early reading skills.\r\n\r\nParents that do not have 
access to a dvd player will have the opportunity to check out a dvd 
player to use for the year.  The plan is to use these videos and edu
cational dvd's for the years to come for other EL students.\r\nnanna
n
==================================================
The 51 fifth grade students that will cycle through my classroom thi
s year all love learning, at least most of the time. At our school, 
97.3% of the students receive free or reduced price lunch. Of the 56
0 students, 97.3% are minority students. \r\nThe school has a vibran
t community that loves to get together and celebrate. Around Hallowe
en there is a whole school parade to show off the beautiful costumes
that students wear. On Cinco de Mayo we put on a big festival with c
rafts made by the students, dances, and games. At the end of the yea
r the school hosts a carnival to celebrate the hard work put in duri
ng the school year, with a dunk tank being the most popular activit
y.My students will use these five brightly colored Hokki stools in p
lace of regular, stationary, 4-legged chairs. As I will only have a 
total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During 
independent reading time they will be used as special chairs student
s will each use on occasion. I will utilize them in place of chairs 
at my small group tables during math and reading times. The rest of 
the day they will be used by the students who need the highest amoun
t of movement in their life in order to stay focused on school.\r\n
\r\nWhenever asked what the classroom is missing, my students always
say more Hokki Stools. They can't get their fill of the 5 stools we 
already have. When the students are sitting in group with me on the 
Hokki Stools, they are always moving, but at the same time doing the
ir work. Anytime the students get to pick where they can sit, the Ho
kki Stools are the first to be taken. There are always students who 
head over to the kidney table to get one of the stools who are disap
pointed as there are not enough of them. \r\n\r\nWe ask a lot of stu
dents to sit for 7 hours a day. The Hokki stools will be a compromis
e that allow my students to do desk work and move at the same time. 
These stools will help students to meet their 60 minutes a day of mo
vement by allowing them to activate their core muscles for balance w
hile they sit. For many of my students, these chairs will take away 
the barrier that exists in schools for a child who can't sit still.n
annan
==================================================
How do you remember your days of school? Was it in a sterile environ
ment with plain walls, rows of desks, and a teacher in front of the 
room? A typical day in our room is nothing like that. I work hard to
create a warm inviting themed room for my students look forward to c
oming to each day.\r\n\r\nMy class is made up of 28 wonderfully uniq
ue boys and girls of mixed races in Arkansas.\r\nThey attend a Title
I school, which means there is a high enough percentage of free and 
reduced-price lunch to qualify. Our school is an \"open classroom\" 
concept, which is very unique as there are no walls separating the c
lassrooms. These 9 and 10 year-old students are very eager learners;
they are like sponges, absorbing all the information and experiences
and keep on wanting more.With these resources such as the comfy red 
throw pillows and the whimsical nautical hanging decor and the blue 
fish nets, I will be able to help create the mood in our classroom s
etting to be one of a themed nautical environment. Creating a classr
oom environment is very important in the success in each and every c
hild's education. The nautical photo props will be used with each ch
ild as they step foot into our classroom for the first time on Meet 
the Teacher evening. I'll take pictures of each child with them, hav
e them developed, and then hung in our classroom ready for their fir
st day of 4th grade.  This kind gesture will set the tone before eve
n the first day of school! The nautical thank you cards will be used
throughout the year by the students as they create thank you cards t
o their team groups.\r\n\r\nYour generous donations will help me to 
help make our classroom a fun, inviting, learning environment from d
ay one.\r\n\r\nIt costs lost of money out of my own pocket on resour
ces to get our classroom ready. Please consider helping with this pr
oject to make our new school year a very successful one. Thank you!n
annan
==================================================
My kindergarten students have varied disabilities ranging from speec
h and language delays, cognitive delays, gross/fine motor delays, to
autism. They are eager beavers and always strive to work their harde
st working past their limitations. \r\n\r\nThe materials we have are
the ones I seek out for my students. I teach in a Title I school whe
re most of the students receive free or reduced price lunch.  Despit
e their disabilities and limitations, my students love coming to sch
ool and come eager to learn and explore.Have you ever felt like you 
had ants in your pants and you needed to groove and move as you were
in a meeting? This is how my kids feel all the time. The want to be 
able to move as they learn or so they say.Wobble chairs are the answ
er and I love then because they develop their core, which enhances g
ross motor and in Turn fine motor skills. \r\nThey also want to lear
n through games, my kids don't want to sit and do worksheets. They w
ant to learn to count by jumping and playing. Physical engagement is
the key to our success. The number toss and color and shape mats can
make that happen. My students will forget they are doing work and ju
st have the fun a 6 year old deserves.nannan
==================================================
My students are a highly mobile population made up of low-income stu
dents and families from a nearby military base. They thrive with a b
alanced approach of high expectations and positive reinforcement, es
pecially as quite a few of my students will attend several schools i
n the elementary years alone. Many of my students struggle in the cl
assroom, but have success in specialized classes like music and art.
I strive to make my art room a space that is creative yet discipline
d, structured yet welcoming - a safe space for everyone.My mission a
s an art teacher is to show my students that art, like all things in
life, is a process. It takes creativity, discipline and perseverance
to make a product that one can take pride in. These life skills will
benefit my students far beyond art.\r\n\r\nMy students need weaving 
```

```
tools like looms, canvas circles, and needles to build patience, fin
e motor skills, discipline and cooperation.\r\nWeaving is a universa
l craft that give me the opportunity to teach about many cultures as
well as make fun stuff with my students. I hope to create several co
operative tapestries with the large looms in hopes that it will \"we
ave us together\" as a community!nannan
==================================================
```

In [28]:

```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 [29]:

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

```
My kindergarten students have varied disabilities ranging from speec
h and language delays, cognitive delays, gross/fine motor delays, to
autism. They are eager beavers and always strive to work their harde
st working past their limitations. \r\n\r\nThe materials we have are
the ones I seek out for my students. I teach in a Title I school whe
re most of the students receive free or reduced price lunch.  Despit
e their disabilities and limitations, my students love coming to sch
ool and come eager to learn and explore.Have you ever felt like you
had ants in your pants and you needed to groove and move as you were
in a meeting? This is how my kids feel all the time. The want to be
able to move as they learn or so they say.Wobble chairs are the answ
er and I love then because they develop their core, which enhances g
ross motor and in Turn fine motor skills. \r\nThey also want to lear
n through games, my kids do not want to sit and do worksheets. They
want to learn to count by jumping and playing. Physical engagement i
s the key to our success. The number toss and color and shape mats c
an make that happen. My students will forget they are doing work and
just have the fun a 6 year old deserves.nannan
==================================================
```

In [30]:

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

```
My kindergarten students have varied disabilities ranging from speec
h and language delays, cognitive delays, gross/fine motor delays, to
autism. They are eager beavers and always strive to work their harde
st working past their limitations.     The materials we have are the
ones I seek out for my students. I teach in a Title I school where m
ost of the students receive 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
ants in your pants and you needed to groove and move as you were in
a meeting? This is how my kids feel all the time. The want to be abl
e to move as they learn or so they say.Wobble chairs are the answer
and I love then because they develop their core, which enhances gros
s motor and in Turn fine motor skills.   They also want to learn thr
ough games, my kids do not want to sit and do worksheets. They want
to learn to count by jumping and playing. Physical engagement is the
key to our success. The number toss and color and shape mats can mak
e that happen. My students will forget they are doing work and just
have the fun a 6 year old deserves.nannan
```

In [31]:

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

```
My kindergarten students have varied disabilities ranging from speec
h and language delays cognitive delays gross fine motor delays to au
tism They are eager beavers and always strive to work their hardest
working past their limitations The materials we have are the ones I
seek out for my students I teach in a Title I school where most of t
he students receive free or reduced price lunch Despite their disabi
lities and limitations my students love coming to school and come ea
ger to learn and explore Have you ever felt like you had ants in you
r pants and you needed to groove and move as you were in a meeting T
his is how my kids feel all the time The want to be able to move as
they learn or so they say Wobble chairs are the answer and I love th
en because they develop their core which enhances gross motor and in
Turn fine motor skills They also want to learn through games my kids
do not want to sit and do worksheets They want to learn to count by
jumping and playing Physical engagement is the key to our success Th
e number toss and color and shape mats can make that happen My stude
nts will forget they are doing work and just have the fun a 6 year o
ld deserves nannan
```

In [32]:

```
# 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', 'itse
lf', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
t', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
s', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becaus
e', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 't
hrough', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al
l', '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'v
e", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "d
idn'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', "should
n't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"}
```

In [35]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    #first convert to lowercase
    sent = (sentance.lower().strip())
    #then remove stop words
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    #now decontract
    sent = decontracted(sent)
    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
    preprocessed_essays.append(sent)
```

```
100%|██████████| 109248/109248 [00:16<00:00, 6653.15it/s]
```

In [36]:

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

Out[36]:

```
'kindergarten students varied disabilities ranging speech language d
elays cognitive delays gross fine motor delays autism eager beavers 
always strive work hardest working past limitations the materials on
es seek students teach title school students receive free reduced pr
ice lunch despite disabilities limitations students love coming scho
ol come eager learn explore have ever felt like ants pants needed gr
oove move meeting kids feel time want able move learn say wobble cha
irs answer love develop core enhances gross motor turn fine motor sk
ills they also want learn games kids want sit worksheets want learn 
count jumping playing physical engagement key success number toss co
lor shape mats make happen students forget work fun 6 year old deser
ves nannan'
```

In [37]:

```
project_data['clean_essay'] = preprocessed_essays
```

In [38]:

```
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','projec
t_essay_4'],axis=1,inplace=True)
```

1.4 Preprocessing of `project_title`

- Decontract project titles, remove line breaks and extra spaces, convert everything to lowercase and then remove all the stop words.

In [42]:

```
preprocessed_titles = []

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

```
100%|██████████| 109248/109248 [00:01<00:00, 56332.93it/s]
```

In [43]:

```
project_data['clean_title'] = preprocessed_titles
project_data.drop(['project_title'],axis=1,inplace=True)
```

Pre-processing teacher_prefix

In [44]:

```
#remove nan from teacher prefix:
#https://stackoverflow.com/questions/21011777/how-can-i-remove-nan-from-list-python-numpy
def remove_nan(prefix):
    if str(prefix)!='nan':
        pr = str(prefix)
        pr = re.sub("\\.","",pr) #remove dot from the end of prefix
        return pr
    return "none"

cleaned_teacher_prefix = project_data['teacher_prefix'].map(remove_nan)
project_data['clean_teacher_prefix'] = cleaned_teacher_prefix
```

In [45]:

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

Pre-process project_grade_category

- Clean the project grade categories:
    - Convert Grades 3-5 ==> Grades_3_5

In [46]:

```
def clean_project_grades(grade):
    grade = re.sub("\-","_",grade)
    grade = re.sub(" ","_",grade)
    return grade.strip()

clean_grades = project_data['project_grade_category'].map(clean_project_grades)
project_data['clean_grade_category'] = clean_grades
```

In [47]:

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

Pre-process project_resource_summary

In [48]:

```
preprocessed_summary = []

for summary in tqdm(project_data['project_resource_summary'].values):
    summary = summary.lower().strip()
    summary = ' '.join(e for e in summary.split() if e.lower() not in stopwords)
    summary = decontracted(summary)
    summary = summary.replace('\\r', ' ')
    summary = summary.replace('\\"', ' ')
    summary = summary.replace('\\n', ' ')
    summary = re.sub('[^A-Za-z0-9]+', ' ', summary)
    preprocessed_summary.append(summary)
```

```
100%|██████████| 109248/109248 [00:03<00:00, 36002.68it/s]
```

In [49]:

```
print(preprocessed_summary[20000])
print(preprocessed_summary[0])
project_data['clean_resource_summary'] = preprocessed_summary
```

```
students need wobble chairs number toss games colors shapes mats mak
e learning fun hands physically engaging
students need opportunities practice beginning reading skills englis
h home
```

In [50]:

```
# Dropping all features we won't need going forward
project_data.drop(['project_resource_summary'],axis=1,inplace=True)
project_data.drop(['Unnamed: 0','teacher_id'],axis=1,inplace=True)
```

In [51]:

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

Out[51]:

	id	school_state	project_submitted_datetime	teacher_number_of_previously_posted_p
0	p253737	IN	2016-12-05 13:43:57	
1	p258326	FL	2016-10-25 09:22:10	

we are going to consider

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

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

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

Assignment 4: Naive Bayes

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

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

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

    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Feature importance**

    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names

4. **Representation of results**

    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
    - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
    - Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

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

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

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

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

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

    - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

2. Naive Bayes

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

In [52]:

```
#Separating features and label column
Y = project_data['project_is_approved']
X = project_data.drop(['project_is_approved','id'],axis=1)
print("Shape of X: ",X.shape)
print("Shape of Y: ",Y.shape)
```

```
Shape of X:  (109248, 13)
Shape of Y:  (109248,)
```

In [53]:

```
#separating data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.30,stratify=
Y)
print("Shape of X_train: ", X_train.shape)
print("Shape of Y_train: ",Y_train.shape)
print("Shape of X_test: ",X_test.shape)
print("Shape of Y_test: ",Y_test.shape)
```

```
Shape of X_train:  (76473, 13)
Shape of Y_train:  (76473,)
Shape of X_test:  (32775, 13)
Shape of Y_test:  (32775,)
```

In [54]:

```
X_train.columns
```

Out[54]:

```
Index(['school_state', 'project_submitted_datetime',
       'teacher_number_of_previously_posted_projects', 'price', 'qua
ntity',
       'clean_categories', 'clean_subcategories', 'essay', 'clean_es
say',
       'clean_title', 'clean_teacher_prefix', 'clean_grade_categor
y',
       'clean_resource_summary'],
      dtype='object')
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 Encoding Categorical Features

One hot encoding: clean_categories

In [55]:

```
from collections import Counter
my_counter = Counter()
for word in X_train['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]))
```

In [56]:

```
# we use count vectorizer to convert the values into one
vectorizer_category = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_category.fit(X_train['clean_categories'].values)

X_train_category_ohe = vectorizer_category.transform(X_train['clean_categories'].values)
X_test_category_ohe = vectorizer_category.transform(X_test['clean_categories'].values)
```

In [57]:

```
print(vectorizer_category.get_feature_names())
print("Shape of X_train after one hot encodig ",X_train_category_ohe.shape)
print("Shape of X_test after one hot encodig ",X_test_category_ohe.shape)
print("Print some random encoded categories: ")
print(X_train_category_ohe[0].toarray())
print(X_test_category_ohe[15].toarray())
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of X_train after one hot encodig  (76473, 9)
Shape of X_test after one hot encodig  (32775, 9)
Print some random encoded categories: 
[[0 0 0 1 0 0 0 0 1]]
[[0 0 0 0 0 0 0 0 1]]
```

One hot encoding: clean_subcategories

In [58]:

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

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

In [59]:

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

X_train_subcategory_ohe = vectorizer_subcategory.transform(X_train['clean_subcategories'].values)
X_test_subcategory_ohe = vectorizer_subcategory.transform(X_test['clean_subcategories'].values)
```

In [60]:

```
print(vectorizer_subcategory.get_feature_names())
print("Shape of X_train subcategory after one hot encodig ",X_train_subcategory_ohe.shape)
print("Shape of X_test subcategory after one hot encodig ",X_test_subcategory_ohe.shape)
print("Print some random encoded categories: ")
print(X_train_subcategory_ohe[0].toarray())
print(X_test_subcategory_ohe[10].toarray())
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Civics_Government', 'Extracurricular', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'EarlyDevelopment', 'Health_LifeScience', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of X_train subcategory after one hot encodig  (76473, 30)
Shape of X_test subcategory after one hot encodig  (32775, 30)
Print some random encoded categories: 
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0]]
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]]
```

One hot encoding: school_state

In [61]:

```
# create a vocabulary for states
unique_states = np.unique(X_train['school_state'].values)

vectorizer_state = CountVectorizer(vocabulary=unique_states,lowercase=False,binary=True)
vectorizer_state.fit(X_train['school_state'].values)

X_train_school_state_ohe = vectorizer_state.transform(X_train['school_state'].values)
X_test_school_state_ohe = vectorizer_state.transform(X_test['school_state'].values)
```

In [62]:

```
print(vectorizer_state.get_feature_names())
print("Shape of X_train school_state after one hot encodig ",X_train_school_stat
e_ohe.shape)
print("Shape of X_test school_state after one hot encodig ",X_test_school_state_
ohe.shape)
print("Print some random encoded school_state: ")
print(X_train_school_state_ohe[0].toarray())
print(X_test_school_state_ohe[15].toarray())
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA',
'HI', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'M
I', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'N
V', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'U
T', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
Shape of X_train school_state after one hot encodig  (76473, 51)
Shape of X_test school_state after one hot encodig  (32775, 51)
Print some random encoded school_state:
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
  0 0 0 0 0 0 0 1 0 0 0 0 0 0 0]]
[[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
  0 0 1 0 0 0 0 0 0 0 0 0 0 0 0]]
```

One hot encoding: teacher_prefix

In [63]:

```
unique_teacher_prefix = np.unique(X_train['clean_teacher_prefix'])

vectorizer_teacher_prefix = CountVectorizer(vocabulary=unique_teacher_prefix,low
ercase=False,binary=True)
vectorizer_teacher_prefix.fit(X_train['clean_teacher_prefix'].values)

X_train_teacher_prefix_ohe = vectorizer_teacher_prefix.transform(X_train['clean_
teacher_prefix'].values)
X_test_teacher_prefix_ohe = vectorizer_teacher_prefix.transform(X_test['clean_te
acher_prefix'].values)
```

In [64]:

```
print(vectorizer_teacher_prefix.get_feature_names())
print("Shape of X_train clean_teacher_prefix after one hot encodig ",X_train_teacher_prefix_ohe.shape)
print("Shape of X_test clean_teacher_prefix after one hot encodig ",X_test_teacher_prefix_ohe.shape)
print("Print some random encoded clean_teacher_prefix: ")
print(X_train_teacher_prefix_ohe[0].toarray())
print(X_test_teacher_prefix_ohe[15].toarray())
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher', 'none']
Shape of X_train clean_teacher_prefix after one hot encodig  (76473,
6)
Shape of X_test clean_teacher_prefix after one hot encodig  (32775,
6)
Print some random encoded clean_teacher_prefix:
[[0 0 0 1 0 0]]
[[0 0 1 0 0 0]]
```

One hot encoding: project_grade_category

In [65]:

```
unique_grades = np.unique(X_train['clean_grade_category'])

vectorizer_grade = CountVectorizer(vocabulary=unique_grades,lowercase=False,binary=True)
vectorizer_grade.fit(X_train['clean_grade_category'].values)


X_train_grade_category_ohe = vectorizer_grade.transform(X_train['clean_grade_category'].values)
X_test_grade_category_ohe = vectorizer_grade.transform(X_test['clean_grade_category'].values)
```

In [66]:

```
print(vectorizer_grade.get_feature_names())
print("Shape of X_train clean_grade_category after one hot encodig ",X_train_grade_category_ohe.shape)
print("Shape of X_test clean_grade_category after one hot encodig ",X_test_grade_category_ohe.shape)
print("Print some random encoded clean_grade_category: ")
print(X_train_grade_category_ohe[0].toarray())
print(X_test_grade_category_ohe[15].toarray())
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of X_train clean_grade_category after one hot encodig  (76473,
4)
Shape of X_test clean_grade_category after one hot encodig  (32775,
4)
Print some random encoded clean_grade_category:
[[0 0 1 0]]
[[0 0 0 1]]
```

2.2.2 Encoding Numerical features

Standardizing Price

In [90]:

```
price_vectorizer = Normalizer().fit(X_train['price'].values.reshape(1,-1))
```

In [91]:

```
X_train_price_normalized = price_vectorizer.transform(X_train['price'].values.reshape(1,-1))
X_test_price_normalized = price_vectorizer.transform(X_test['price'].values.reshape(1, -1))
```

Standardize teacher_number_of_previously_posted_projects

In [95]:

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

In [96]:

```
X_train_normal_previous_project = price_vectorizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_test_normal_previous_project = price_vectorizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
```

2.3 Make Data Model Ready: encoding eassay, and project_title

2.3.1 Bag of words : Essay

In [97]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_essay_bow = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=5000)
vectorizer_essay_bow.fit(X_train['clean_essay'])
```

Out[97]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [98]:

```
X_train_essay_bow = vectorizer_essay_bow.transform(X_train['clean_essay'])
X_test_essay_bow = vectorizer_essay_bow.transform(X_test['clean_essay'])

print("Shape of X_train_essay_bow ",X_train_essay_bow.shape)
print("Shape of X_test_essay_bow ",X_test_essay_bow.shape)
```

```
Shape of X_train_essay_bow  (76473, 5000)
Shape of X_test_essay_bow  (32775, 5000)
```

2.3.2 Bag of words : Project Title

In [99]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_title_bow = CountVectorizer(min_df=10,ngram_range=(1,2), max_features
=5000)
vectorizer_title_bow.fit(X_train['clean_title'])
```

Out[99]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='stric
t',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='conten
t',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [100]:

```
X_train_title_bow = vectorizer_title_bow.transform(X_train['clean_title'])
X_test_title_bow = vectorizer_title_bow.transform(X_test['clean_title'])

print("Shape of X_train_title_bow ",X_train_title_bow.shape)
print("Shape of X_test_title_bow ",X_test_title_bow.shape)
```

```
Shape of X_train_title_bow  (76473, 4861)
Shape of X_test_title_bow  (32775, 4861)
```

2.3.3 TFIDF vectorizer: Essay

In [101]:

```
vectorizer_essay_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,2), max_featur
es=5000)
vectorizer_essay_tfidf.fit(X_train['clean_essay'])
```

Out[101]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='stric
t',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='cont
ent',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), norm='l2', preprocessor=None, smooth_idf
=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=T
rue,
        vocabulary=None)
```

In [102]:

```
X_train_essay_tfidf = vectorizer_essay_tfidf.transform(X_train['clean_essay'])
X_test_essay_tfidf = vectorizer_essay_tfidf.transform(X_test['clean_essay'])

print("Shape of X_train_essay_tfidf ",X_train_essay_tfidf.shape)
print("Shape of X_test_essay_tfidf ",X_test_essay_tfidf.shape)
```

```
Shape of X_train_essay_tfidf  (76473, 5000)
Shape of X_test_essay_tfidf  (32775, 5000)
```

2.3.4 TFIDF vectorizer: Project title

In [103]:

```
vectorizer_title_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,2), max_featur
es=5000)
vectorizer_title_tfidf.fit(X_train['clean_title'])
```

Out[103]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='stric
t',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='cont
ent',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), norm='l2', preprocessor=None, smooth_idf
=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=T
rue,
        vocabulary=None)
```

In [104]:

```
X_train_title_tfidf = vectorizer_title_tfidf.transform(X_train['clean_title'])
X_test_title_tfidf = vectorizer_title_tfidf.transform(X_test['clean_title'])

print("Shape of X_train_title_tfidf ",X_train_title_tfidf.shape)
print("Shape of X_test_title_tfidf",X_test_title_tfidf.shape)
```

```
Shape of X_train_title_tfidf  (76473, 4861)
Shape of X_test_title_tfidf (32775, 4861)
```

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

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

2.4.1 Applying Naive Bayes on BOW, SET 1

In [105]:

```
f1 = X_train_school_state_ohe
f2 = X_train_category_ohe
f3 = X_train_subcategory_ohe
f4 = X_train_grade_category_ohe
f5 = X_train_teacher_prefix_ohe
f6 = np.array(X_train_price_normalized.reshape(-1,1))
f7 = np.array(X_train_normal_previous_project.reshape(-1,1))

X_train_nb = hstack((f1,f2,f3,f4,f5,f6,f7,X_train_essay_bow,X_train_title_bow))
```

In [106]:

```
X_train_nb.shape
```

Out[106]:

```
(76473, 9963)
```

In [107]:

```
f1 = X_test_school_state_ohe
f2 = X_test_category_ohe
f3 = X_test_subcategory_ohe
f4 = X_test_grade_category_ohe
f5 = X_test_teacher_prefix_ohe
f6 = np.array(X_test_price_normalized).reshape(-1,1)
f7 = np.array(X_test_normal_previous_project).reshape(-1,1)

X_test_nb = hstack((f1,f2,f3,f4,f5,f6,f7,X_test_essay_bow,X_test_title_bow))
X_test_nb.shape
```

Out[107]:

```
(32775, 9963)
```

In [108]:

```
multinomial_nb = MultinomialNB(class_prior=[0.5,0.5])
#Set parameters for grid search
parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.0
5, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]}
# Use GridSearchCV to search for the optimal value of alpha
# Here, we are using roc_auc as our scoring metric since we have imbalanced data
set
clf = GridSearchCV(estimator = multinomial_nb, param_grid = parameters, cv=3, sc
oring='roc_auc', return_train_score=True, n_jobs=-1, verbose = True)
#pass X_train and Y_train as data to search alpha. Here grid search will automat
ically split the data
#into stratified samples.
clf.fit(X_train_nb, Y_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent w
orkers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   10.9s
[Parallel(n_jobs=-1)]: Done  60 out of  60 | elapsed:   13.8s finish
ed
```

Out[108]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=MultinomialNB(alpha=1.0, class_prior=[0.5, 0.5], fi
t_prior=True),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'alpha': [1e-05, 5e-05, 0.0001, 0.0005, 0.001, 0.
005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000,
10000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='roc_auc', verbose=True)
```

In [109]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
results.head()
```

Out[109]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split
0	0.426653	0.007295	0.072742	0.000015	1e-05	{'alpha': 1e-05}	
1	0.379483	0.037526	0.061389	0.007808	5e-05	{'alpha': 5e-05}	
2	0.365641	0.015588	0.060348	0.005860	0.0001	{'alpha': 0.0001}	
3	0.351679	0.006078	0.064404	0.008471	0.0005	{'alpha': 0.0005}	
4	0.390828	0.022623	0.062599	0.006368	0.001	{'alpha': 0.001}	

In [110]:

```
clf.best_estimator_
```

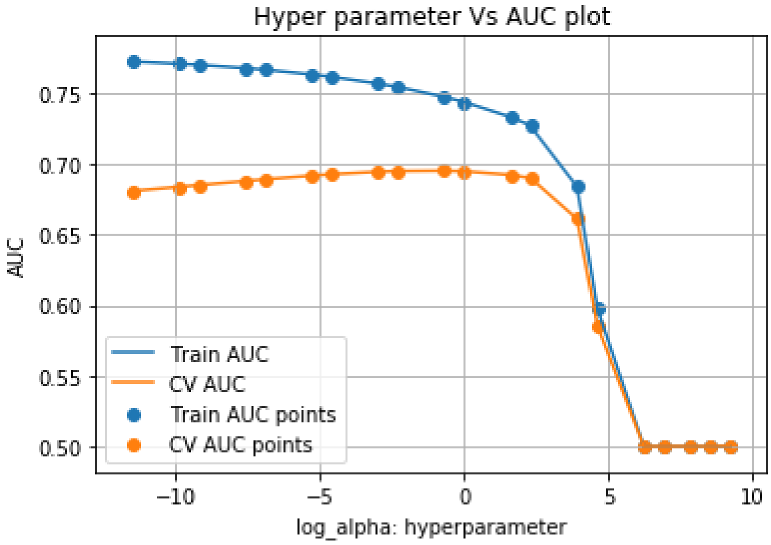
Out[110]:

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

Plotting alpha vs train and CV error

In [111]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_alpha'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
alphas =  results['param_alpha']

log_alphas = [np.log(x) for x in alphas]

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039

plt.plot(log_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039

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("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()

results.head()
```



Out[111]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split
0	0.426653	0.007295	0.072742	0.000015	1e-05	{'alpha': 1e-05}	
1	0.379483	0.037526	0.061389	0.007808	5e-05	{'alpha': 5e-05}	
2	0.365641	0.015588	0.060348	0.005860	0.0001	{'alpha': 0.0001}	
3	0.351679	0.006078	0.064404	0.008471	0.0005	{'alpha': 0.0005}	
4	0.390828	0.022623	0.062599	0.006368	0.001	{'alpha': 0.001}	

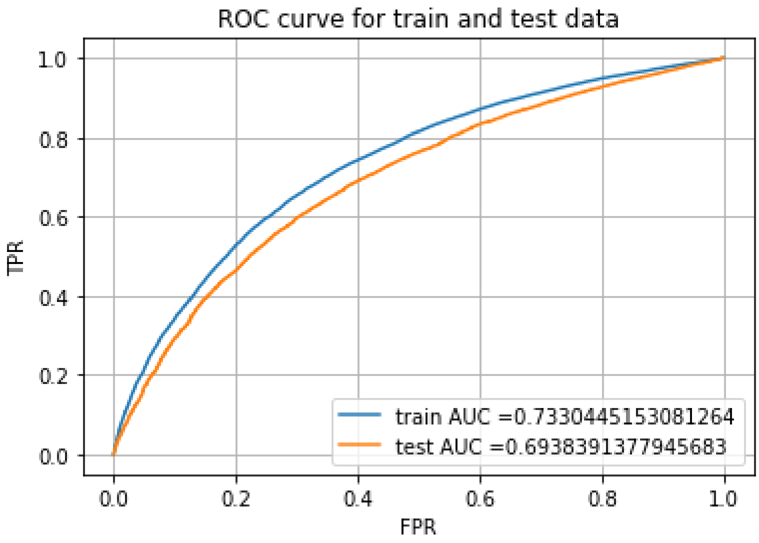
Observations

1. For very large values of alpha, the model performs poorly on train and CV data.
2. For very small values of alpha, model performs well on train data but not on CV data.
3. At alpha = 0.5, model has the best performance on train and CV set.

Training model with hyperparameter alpha = 0.5

In [113]:

```
multinomial_nb_bow = MultinomialNB(alpha=0.5,class_prior=[0.5,0.5])
multinomial_nb_bow.fit(X_train_nb, Y_train)

y_train_pred = multinomial_nb_bow.predict_proba(X_train_nb)
y_test_pred = multinomial_nb_bow.predict_proba(X_test_nb)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
es of the positive class
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_test_pred[:,1])

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr
)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()

plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



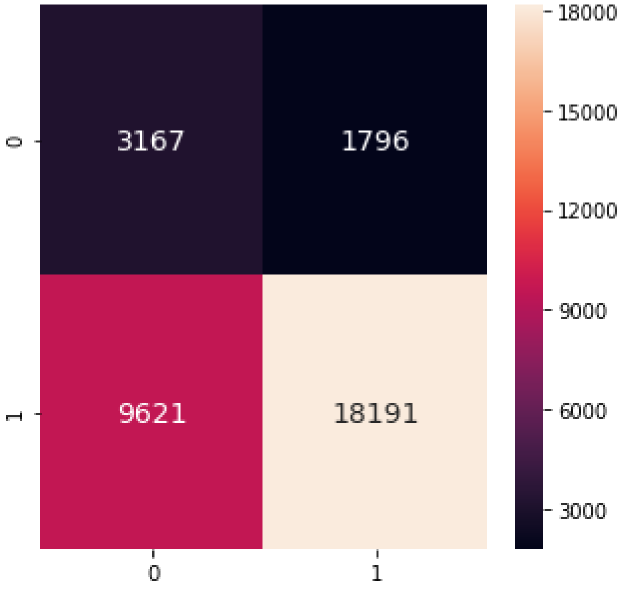
Confusion Matrix

In [114]:

```
y_test_predict = multinomial_nb_bow.predict(X_test_nb)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe08dfd12e8>
```



2.4.1.1 Top 10 important features of positive class from SET 1

- Create a list of all the features that have been identified by the One hot and BOW vectorization.

In [121]:

```
# this list will contain all the features identified in our trainind data
# with bow vectorization of title and essay
all_features = []

for feature in vectorizer_state.get_feature_names():
    all_features.append(feature)

for feature in vectorizer_category.get_feature_names():
    all_features.append(feature)

for feature in vectorizer_subcategory.get_feature_names():
    all_features.append(feature)

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

for feature in vectorizer_teacher_prefix.get_feature_names():
    all_features.append(feature)

all_features.append("price")
all_features.append("project")

for feature in vectorizer_essay_bow.get_feature_names():
    all_features.append(feature)

for feature in vectorizer_title_bow.get_feature_names():
    all_features.append(feature)
#Features we collected are equal to the actual features.
print("Toatal features: ", len(all_features))
print("Available features during training: ", X_train_nb.shape[1])
```

```
Toatal features:  9963
Available features during training:  9963
```

In [122]:

```
df = pd.DataFrame( {'feature_name':all_features, 'log_probability':multinomial_n
b_bow.feature_log_prob_[1]})
a = df.sort_values(by=['log_probability'], ascending=False)
a.head(10)
```

Out[122]:

	feature_name	log_probability
4214	students	-3.167895
3777	school	-4.312141
2498	learning	-4.677809
834	classroom	-4.703038
2445	learn	-5.017440
2048	help	-5.044304
2770	many	-5.187343
2977	nannan	-5.203731
3085	not	-5.204054
2994	need	-5.317682

2.4.1.2 Top 10 important features of negative class from SET 1

In [123]:

```
df_nega = pd.DataFrame( {'feature_name':all_features, 'log_probability':multinom
ial_nb_bow.feature_log_prob_[0]})
b = df_nega.sort_values(by=['log_probability'], ascending=False)
b.head(10)
```

Out[123]:

	feature_name	log_probability
4214	students	-3.184938
3777	school	-4.276292
2498	learning	-4.603877
834	classroom	-4.754119
2445	learn	-4.941562
2048	help	-4.997844
2977	nannan	-5.151252
3085	not	-5.171054
2770	many	-5.183761
2994	need	-5.293763

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [124]:

```
f1 = X_train_school_state_ohe
f2 = X_train_category_ohe
f3 = X_train_subcategory_ohe
f4 = X_train_grade_category_ohe
f5 = X_train_teacher_prefix_ohe
f6 = np.array(X_train_price_normalized).reshape(-1,1)
f7 = np.array(X_train_normal_previous_project).reshape(-1,1)

X_train_nb = hstack((f1,f2,f3,f4,f5,f6,f7,X_train_essay_tfidf,X_train_title_tfid
f))
```

In [125]:

```
X_train_nb.shape
```

Out[125]:

```
(76473, 9963)
```

In [126]:

```
f1 = X_test_school_state_ohe
f2 = X_test_category_ohe
f3 = X_test_subcategory_ohe
f4 = X_test_grade_category_ohe
f5 = X_test_teacher_prefix_ohe
f6 = np.array(X_test_price_normalized).reshape(-1,1)
f7 = np.array(X_test_normal_previous_project).reshape(-1,1)

X_test_nb = hstack((f1,f2,f3,f4,f5,f6,f7,X_test_essay_tfidf,X_test_title_tfidf))
X_test_nb.shape
```

Out[126]:

```
(32775, 9963)
```

In [127]:

```
multinomial_nb = MultinomialNB(class_prior=[0.5,0.5])
#Set parameters for grid search
parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.0
5, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]}
# Use GridSearchCV to search for the optimal value of alpha
# Here, we are using roc_auc as our scoring metric since we have imbalanced data
set
clf = GridSearchCV(estimator = multinomial_nb, param_grid = parameters, cv=10, s
coring='roc_auc', return_train_score=True, n_jobs=-1, verbose = True)
#pass X_train and Y_train as data to search alpha. Here grid search will automat
ically split the data
#into stratified samples.
clf.fit(X_train_nb, Y_train)
```

```
Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent w
orkers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   16.2s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   38.3s
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:   39.4s finish
ed
```

Out[127]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=MultinomialNB(alpha=1.0, class_prior=[0.5, 0.5], fi
t_prior=True),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'alpha': [1e-05, 5e-05, 0.0001, 0.0005, 0.001, 0.
005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000,
10000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='roc_auc', verbose=True)
```

In [130]:

```
clf.best_estimator_
```

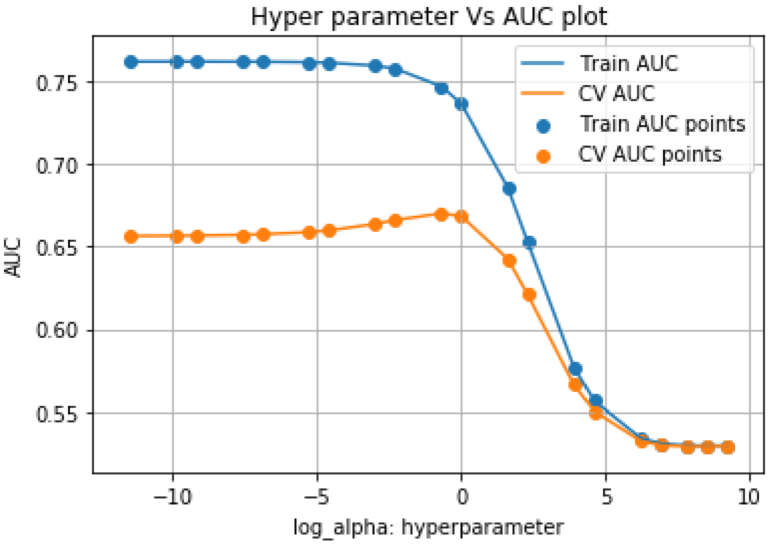
Out[130]:

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

Plotting alpha vs train and CV error

In [131]:

```
results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_alpha'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
alphas =  results['param_alpha']

log_alphas = [np.log(x) for x in alphas]

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039

plt.plot(log_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039

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("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()

results.head()
```



```
Out[131]:
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split
0	0.471856	0.087617	0.018326	0.001587	1e-05	{'alpha': 1e-05}	
1	0.390843	0.009914	0.019167	0.001617	5e-05	{'alpha': 5e-05}	
2	0.441101	0.054464	0.022687	0.004653	0.0001	{'alpha': 0.0001}	
3	0.435339	0.053271	0.020261	0.002416	0.0005	{'alpha': 0.0005}	
4	0.418580	0.026921	0.019158	0.002312	0.001	{'alpha': 0.001}	

5 rows × 31 columns

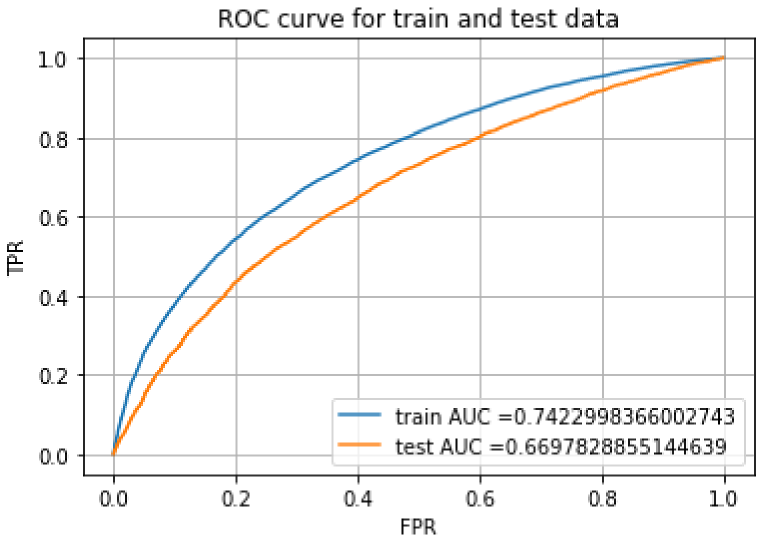
Observations

1. For very large values of alpha, the model performs poorly on train and CV data.
2. For very small values of alpha, model performs well on train data but not on CV data.
3. At alpha = 0.5, model has the best performance on train and CV set.

Training model with hyperparameter alpha = 0.5

In [132]:

```
multinomial_nb_tfidf = MultinomialNB(alpha=0.5,class_prior=[0.5,0.5])
multinomial_nb_tfidf.fit(X_train_nb, Y_train)

y_train_pred = multinomial_nb_tfidf.predict_proba(X_train_nb)
y_test_pred = multinomial_nb_tfidf.predict_proba(X_test_nb)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_test_pred[:,1])

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()

plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



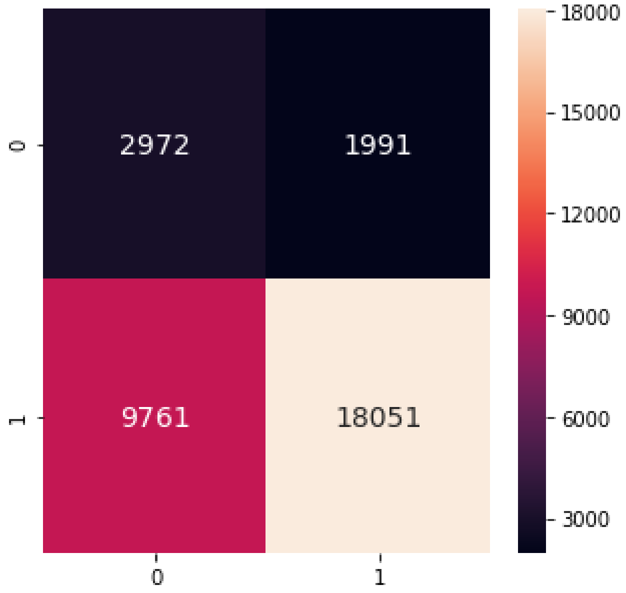
Confusion Matrix

In [133]:

```
y_test_predict = multinomial_nb_tfidf.predict(X_test_nb)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[133]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe08e3a2278>
```



2.4.2.1 Top 10 important features of positive class from SET 2

- Create a list of all the features that have been identified by the One hot and TFIDF vectorization.

In [134]:

```
# this list will contain all the features identified in our trainind data
# with tfidf vectorization of title and essay
all_features_tfidf = []

for feature in vectorizer_state.get_feature_names():
    all_features_tfidf.append(feature)

for feature in vectorizer_category.get_feature_names():
    all_features_tfidf.append(feature)

for feature in vectorizer_subcategory.get_feature_names():
    all_features_tfidf.append(feature)

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

for feature in vectorizer_teacher_prefix.get_feature_names():
    all_features_tfidf.append(feature)

all_features_tfidf.append("price")
all_features_tfidf.append("project")

for feature in vectorizer_essay_tfidf.get_feature_names():
    all_features_tfidf.append(feature)

for feature in vectorizer_title_tfidf.get_feature_names():
    all_features_tfidf.append(feature)

#The features we collected are equal to the actual features
print("Toatal features: ", len(all_features_tfidf))
print("Available features during training: ", X_train_nb.shape[1])
```

```
Toatal features:  9963
Available features during training:  9963
```

In [135]:

```
df = pd.DataFrame( {'feature_name':all_features_tfidf, 'log_probability':multinomial_nb_tfidf.feature_log_prob_[1]})
a = df.sort_values(by=['log_probability'], ascending=False)
a.head(10)
```

Out[135]:

	feature_name	log_probability
96	Mrs	-3.522996
59	Literacy_Language	-3.602071
93	Grades_PreK_2	-3.793050
58	Math_Science	-3.864216
97	Ms	-3.917765
90	Grades_3_5	-3.955651
89	Literacy	-4.034415
88	Mathematics	-4.249729
87	Literature_Writing	-4.464485
91	Grades_6_8	-4.753901

2.4.2.2 Top 10 important features of negative class from SET 2

In [136]:

```
df_nega = pd.DataFrame( {'feature_name':all_features_tfidf, 'log_probability':multinomial_nb_tfidf.feature_log_prob_[0]})
b = df_nega.sort_values(by=['log_probability'], ascending=False)
b.head(10)
```

Out[136]:

	feature_name	log_probability
96	Mrs	-3.577353
59	Literacy_Language	-3.742677
93	Grades_PreK_2	-3.790359
58	Math_Science	-3.807060
97	Ms	-3.898567
90	Grades_3_5	-4.012734
88	Mathematics	-4.228577
89	Literacy	-4.246509
87	Literature_Writing	-4.539510
91	Grades_6_8	-4.725567

3. Conclusions

In [177]:

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

x.add_row(["BOW", "Naive Bayes", 0.5, 0.69])
x.add_row(["TFIDF", "Naive Bayes", 0.5, 0.67])

print(x)
```

```
+------------+-------------+-----------------------+------+
| Vectorizer |    Model    | Alpha:Hyper Parameter | AUC  |
+------------+-------------+-----------------------+------+
|    BOW     | Naive Bayes |          0.5          | 0.69 |
|   TFIDF    | Naive Bayes |          0.5          | 0.67 |
+------------+-------------+-----------------------+------+
```

Summary

1. Naive Bayes performs equally well on both BOW and TFIDF encodings, with a little improvement when tfidf is used.
2. Naive Bayes takes significantly less time to train on the entire dataset than KNN.
3. Naive Bayes performs much better than KNN, this can be concluded from our computations done before.