

DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
<code>project_id</code>	A unique identifier for the proposed project. Example: p036502
<code>project_title</code>	Title of the project. Examples: <ul style="list-style-type: none">Art Will Make You Happy!First Grade Fun
<code>project_grade_category</code>	Grade level of students for which the project is targeted. One of the following enumerated values: <ul style="list-style-type: none">Grades PreK-2Grades 3-5Grades 6-8Grades 9-12
<code>project_subject_categories</code>	One or more (comma-separated) subject categories for the project from the following enumerated list of values: <ul style="list-style-type: none">Applied LearningCare & HungerHealth & SportsHistory & CivicsLiteracy & LanguageMath & ScienceMusic & The ArtsSpecial NeedsWarmth Examples: <ul style="list-style-type: none">Music & The ArtsLiteracy & 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: <ul style="list-style-type: none"> Literacy Literature & Writing, Social Sciences
project_resource_summary	An explanation of the resources needed for the project. Example: <ul style="list-style-type: none"> My students need hands on literacy materials to manage sensory needs!
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: <ul style="list-style-type: none"> 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
<code>id</code>	A <code>project_id</code> value from the <code>train.csv</code> file. Example: <code>p036502</code>
<code>description</code>	Description of the resource. Example: <code>Tenor Saxophone Reeds, Box of 25</code>
<code>quantity</code>	Quantity of the resource required. Example: <code>3</code>
<code>price</code>	Price of the resource required. Example: <code>9.95</code>

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

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

Label	Description
<code>project_is_approved</code>	A binary flag indicating whether DonorsChoose approved the project. A value of <code>0</code> indicates the project was not approved, and a value of <code>1</code> 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 [10]:

```
%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 [11]:

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

In [12]:

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

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

```
-----
----
```

```
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects'
 'project_is_approved']
```

In [13]:

```
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[13]:

	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 [14]:

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

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

            j=j.replace('The', '') # if we have the words "The
" we are going to replace it with ''(i.e removing 'The')
            j = j.replace(' ', '') # we are placing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
            temp+=j.strip()+" " # " abc ".strip() will return "abc
", remove the trailing spaces
            temp = temp.replace('&', '_') # we are replacing the &
value into
```

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

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

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

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

1.3 preprocessing of project_subject_subcategories

In [15]:

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

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

sub_cat_list = []
for i in sub_categories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth , Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the category based on space "Math & Science"=> "Math", "&", "Science"

        j=j.replace('The', '') # if we have the words "The " we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ', '') # we are placing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" #" abc ".strip() will return "abc", remove the trailing spaces
    temp = temp.replace('&', '_')
```

```
sub_cat_list.append(temp.strip())

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

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

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

preprocessing of teacher_prefix

In [16]:

```
#NaN values in teacher prefix will create a problem while encoding, so we replace NaN values with the mode of that particular 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 [17]:

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

In [18]:

```
project_data.drop(['teacher_prefix'], axis = 1, inplace = True)  
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[18]:

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

preprocessing of project_grade_category

In [19]:

```
# 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)
```

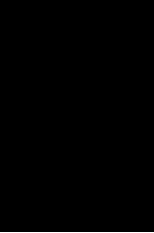
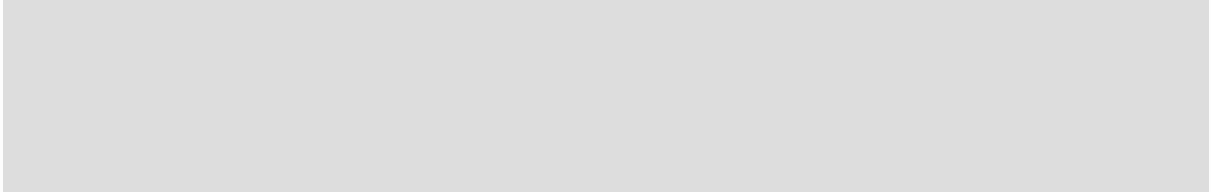
In [20]:

```
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[20]:

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



1.3 Text preprocessing

In [21]:

```
# 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 [22]:

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

Out[22]:

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



In [23]:

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

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 countries represented with the families within our school.

Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein Our English learner's have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English along side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one at home is able to assist. All fami

lies with students within the Level 1 proficiency status, will be offered to be a part of this program. These educational videos will be specially chosen by the English Learner Teacher and will be sent home regularly to watch.

The videos are to help the child develop early reading skills.
Parents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.
nnannan

=====
====

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students.
The school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity. My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will ut

ilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the students who need the highest amount of movement in their life in order to stay focused on school.

Whenever asked what the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them.

We ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still.

=====
=====

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

My class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas. They attend a Title I sc

hool, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an "open classroom" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more. With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.

Your generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.

It costs a lot of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!

nannan

=====
=====

My kindergarten students have varied disabilities ranging from speech and language delays, 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. They want to be able to
move as they learn or so they say. Wobble chai
rs are the answer and I love them 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 am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character. In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.

The cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use. The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.

nannan
=====

In [24]:

```
# 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 [25]:

```

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

```

My students are a highly mobile population made up of low-income students and families from a nearby military base. They thrive with a balanced approach of high expectations and positive reinforcement, especially as quite a few of my students will attend several schools in the elementary years alone. Many of my students struggle in the classroom, but have success in specialized classes like music and art. I strive to make my art room a space that is creative yet disciplined, structured yet welcoming - a safe space for everyone. My mission as 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.

My students need weaving tools like looms, canvas circles, and needles to build patience, fine motor skills, discipline and cooperation.

Weaving is a universal 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 cooperative tapestries with the large looms in hopes that it will "weave us together" as a community!

nnan

=====
=====

In [26]:

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

My students are a highly mobile population made up of low-income students and families from a nearby military base. They thrive with a balanced approach of high expectations and positive reinforcement, especially as quite a few of my students will attend several schools in the elementary years alone. Many of my students struggle in the classroom, but have success in specialized classes like music and art. I strive to make my art room a space that is creative yet disciplined, structured yet welcoming - a safe space for everyone. My mission as 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. My students need weaving tools like looms, canvas circles, and needles to build patience, fine motor skills, discipline and cooperation. Weaving is a universal craft that gives me the opportunity to teach about many cultures 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 [27]:

```
#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 made up of low income students and families from a nearby military base They thrive with a balanced approach of high expectations and positive reinforcement especially as quite a few of my students will attend several schools in the elementary years alone Many of my students struggle in the classroom but have success in specialized classes like music and art I strive to make my art room a space that is creative yet disciplined structured yet welcoming a safe space for everyone My mission as 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 My students need weaving tools like looms canvas circles and needles to build patience fine motor skills discipline and cooperation Weaving is a universal 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 cooperative tapestries with the large looms in hopes that it will weave us together as a community

In [28]:

```
# 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', 'b
etween', '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', "co
uldn'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 [29]:

```
#convert all the words to lower case first and then remove th
```

```
e stopwords
for i in range(len(project_data['essay'].values)):
    project_data['essay'].values[i] = project_data['essay'].values[i].lower()
```

In [30]:

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

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

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

```
100%|██████████| 109248/109248 [00:44<00:00, 2467.28it/s]
```

In [31]:

```
# after preprocessing
preprocessed_essays[20000]
```

Out[31]:

```
'kindergarten students varied disabilities ranging speech language delays cognitive delays g
```

ross fine motor delays autism eager beavers al
ways strive work hardest working past limitati
ons materials ones seek students teach title s
chool students receive free reduced price lunc
h despite disabilities limitations students lo
ve coming school come eager learn explore ever
felt like ants pants needed groove move meeti
ng kids feel time want able move learn say wob
ble chairs answer love develop core enhances g
ross motor turn fine motor skills also want le
arn games kids not want sit worksheets want le
arn count jumping playing physical engagement
key success number toss color shape mats make
happen students forget work fun 6 year old des
erves'

In [32]:

```
#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)
```

Preprocessing of $project_{it} \leq$

In [33]:

```
#convert all the words to lower case first and then remove the stopwords
for i in range(len(project_data['project_title'].values)):
    project_data['project_title'].values[i] = project_data['project_title'].values[i].lower()
```

In [34]:

```
# 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('\n', ' ')
    sent = sent.replace('\n', ' ')
    sent = sent.replace('nan', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:01<00:00, 59188.75it/s]
```

In [35]:

```
preprocessed_titles[20000]
```

Out[35]:

'need move input'

In [36]:

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

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

In [37]:

```
# train test split

from sklearn.model_selection import 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'])
```

In [38]:

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

print('Test dataset:', len(project_data_test)/len(project_data)*100, '%\n', 'size:', len(project_data_test))
```

Split ratio

Train dataset: 66.99985354422964 %
size: 73196

Test dataset: 33.000146455770356 %
size: 36052

In [39]:

```
#Features
```



```
project_data_train.drop(['project_is_approved'], axis=1, inplace=True)
```

```
project_data_test.drop(['project_is_approved'], axis=1, inplace=True)
```

1.5 Preparing data for models

In [40]:

```
project_data.columns
```

Out[40]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',  
      'project_submitted_datetime', 'project_title',  
      'project_resource_summary',  
      'teacher_number_of_previously_posted_projects', 'project_is_approved',  
      'clean_categories', 'clean_subcategories', 'teacher_prefix',  
      'project_grade_category', 'essay', 'preprocessed_essays',  
      'preprocessed_titles'],  
      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 : numerical

- 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 [41]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(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)

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 test data matrix after one hot encoding ",
      test_categories_one_hot.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language', 'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
```

Shape of training data matrix after one hot encoding (73196, 9)

Shape of test data matrix after one hot encoding (36052, 9)

In [43]:

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

train_subcategories_one_hot = vectorizer_subcat.transform(project_data_train['clean_subcategories'].values)

test_subcategories_one_hot = vectorizer_subcat.transform(project_data_test['clean_subcategories'].values)

print(vectorizer_subcat.get_feature_names())

print("Shape of train data matrix after one hot encoding ", train_subcategories_one_hot.shape)

print("Shape of test data matrix after one hot encoding ", test_subcategories_one_hot.shape)
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government', 'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevel
```

```
opment', 'Economics', 'EnvironmentalScience',  
'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness', 'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'Mathematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'SocialSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
```

```
Shape of train data matrix after one hot encoding (73196, 30)
```

```
Shape of test data matrix after one hot encoding (36052, 30)
```

In [44]:

```
## 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)  
  
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 ",  
train_school_state_category_one_hot.shape)
```

```
print("Shape of test data matrix after one hot encoding ",test_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 encoding (73196, 51)
```

```
Shape of test data matrix after one hot encoding (36052, 51)
```

In [46]:

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

In [47]:

```
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 [48]:

```
## we use count vectorizer to convert the values into one hot encoded features  
  
vectorizer_grade = CountVectorizer(lowercase=False, binary=True)
```

```
vectorizer_grade.fit(project_data_train['project_grade_category'].values)
```

```
print(vectorizer_grade.get_feature_names())
```

```
train_project_grade_category_one_hot = vectorizer_grade.transform(project_data_train['project_grade_category'].values)
```

```
test_project_grade_category_one_hot = vectorizer_grade.transform(project_data_test['project_grade_category'].values)
```

```
print("Shape of train data matrix after one hot encoding ", train_project_grade_category_one_hot.shape)
```

```
print("Shape of test data matrix after one hot encoding ", test_project_grade_category_one_hot.shape)
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
```

```
Shape of train data matrix after one hot encoding (73196, 4)
```

```
Shape of test data matrix after one hot encoding (36052, 4)
```

In [49]:

```
#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'].values.astype("U"))
```



```
print(vectorizer_prefix.get_feature_names())

train_teacher_prefix_categories_one_hot = vectorizer_prefix.trans
form(project_data_train['teacher_prefix'].values.astype("U"))

test_teacher_prefix_categories_one_hot = vectorizer_prefix.trans
form(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 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 encoding (73196, 5)
Shape of test data matrix after one hot encoding (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 [50]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(project_data_train['preprocessed_essays'].values) #Fitting has to be on Train data

train_essay_bow = vectorizer_bow_essay.transform(project_data_train['essay'].values)

test_essay_bow = vectorizer_bow_essay.transform(project_data_test['essay'].values)

print("Shape of train data matrix after one hot encoding ", train_essay_bow.shape)

print("Shape of test data matrix after one hot encoding ", test_essay_bow.shape)
```

Shape of train data matrix after one hot encod

```
ing (73196, 14091)
Shape of test data matrix after one hot encodi
ng (36052, 14091)
```

In [51]:

```
# 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)

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 test data matrix after one hot encoding ",tes
t_title_bow.shape)
```

```
Shape of train data matrix after one hot encod
ing (73196, 2529)
Shape of test data matrix after one hot encodi
ng (36052, 2529)
```

1.5.2.2 TFIDF vectorizer

In [52]:

```

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)

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 test data matrix after one hot encoding ", test_essay_tfidf.shape)

```

Shape of train data matrix after one hot encoding (73196, 14091)
 Shape of test data matrix after one hot encoding (36052, 14091)

In [53]:

```

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)

test_title_tfidf = vectorizer_tfidf_title.transform(project_data_test['preprocessed_titles'].values)

print("Shape of train data matrix after one hot encoding ", train_title_tfidf.shape)

```

```
print("Shape of test data matrix after one hot encoding ",test_title_tfidf.shape)
```

Shape of train data matrix after one hot encoding (73196, 2529)

Shape of test data matrix after one hot encoding (36052, 2529)

1.5.2.3 Using Pretrained Models: Avg W2V

In [54]:

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

In [55]:

```
# 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%|██████████| 73196/73196 [00:12<00:00, 606
8.82it/s]

```

```

73196
300

```

In [56]:

```

# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_essays = []; # the avg-w2v for each sentence/review
                           # 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 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_essays.append(vector)

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

```

100%|██████████| 36052/36052 [00:05<00:00, 607
0.31it/s]

36052

300

In [57]:

```
# 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%|██████████| 73196/73196 [00:00<00:00, 981
03.09it/s]

73196

300

In [58]:

```

# 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, 100
379.74it/s]

```

```

36052

```

```

300

```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [59]:

```

# 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(
tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())

```

In [60]:

```

# 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%|██████████| 73196/73196 [01:26<00:00, 850  
.60it/s]
```

```
73196
```

```
300
```

In [61]:

```
# 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 [00:42<00:00, 851
.85it/s]
```

```
36052
```

```
300
```

In [62]:

```
# 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(
tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [63]:

```
# 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[wo
```

```
rd]) 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%|██████████| 73196/73196 [00:01<00:00, 472
20.82it/s]
```

```
73196
```

```
300
```

In [64]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_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
    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 word
```

```

        # 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
        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, 489
15.02it/s]

```

```

36052

```

```

300

```

1.5.3 Vectorizing Numerical features

In [65]:

```

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

```

In [66]:

```

project_data_train = pd.merge(project_data_train, price_data,
on='id', how='left')

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

```

In [67]:

```
from sklearn.preprocessing import Normalizer
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# 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_train['price'].values.reshape(1, -1))

price_normalized_test = normalizer.transform(project_data_test['price'].values.reshape(1, -1))
#reshaping again after normalization

price_normalized_train = price_normalized_train.reshape(-1, 1)
price_normalized_test = price_normalized_test.reshape(-1, 1)

print('After normalization')
print(price_normalized_train.shape)

print(price_normalized_test.shape)
```

After normalization
(73196, 1)
(36052, 1)

In [68]:

```
normalizer = Normalizer()
```

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

```
previously_posted_projects_normalized_train = normalizer.transform(project_data_train['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))
```

```
#reshaping again after normalization
```

```
previously_posted_projects_normalized_train = previously_posted_projects_normalized_train.reshape(-1,1)
```

```
previously_posted_projects_normalized_test = previously_posted_projects_normalized_test.reshape(-1,1)
```

```
print('After normalization')
```

```
print(previously_posted_projects_normalized_train.shape)
```

```
print(previously_posted_projects_normalized_test.shape)
```

After normalization

(73196, 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)

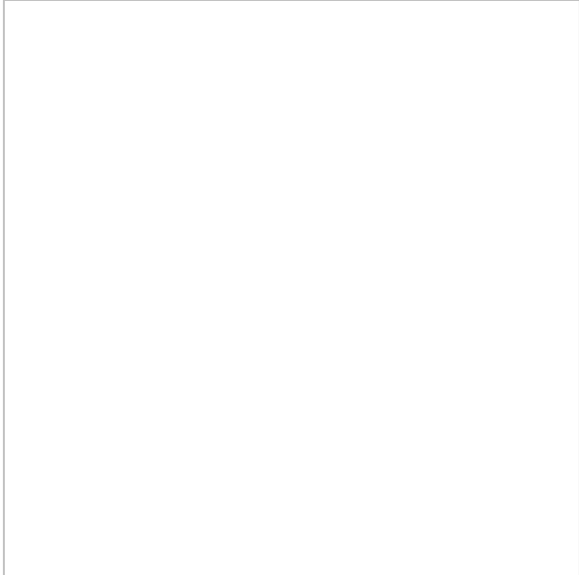
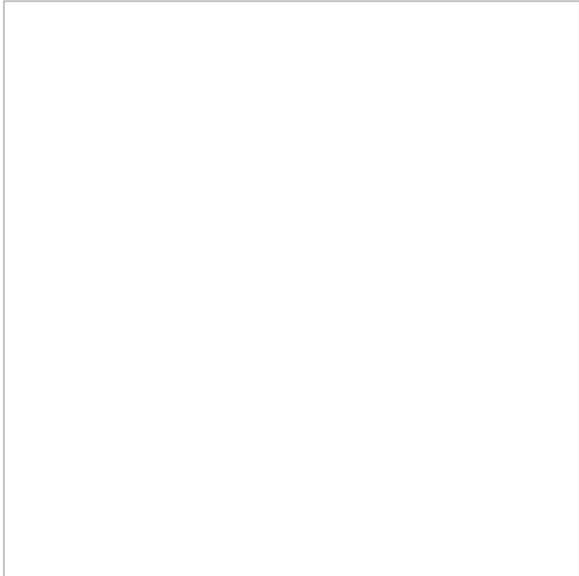
- Find the best hyper parameter which will give the maximum [AUC](#) value
- Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
- Find the best hyper paramter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or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

3. Feature importance

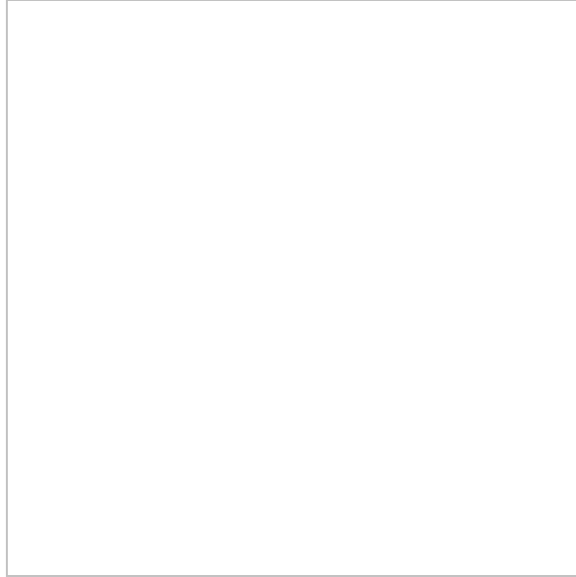
- Find the top 10 features of positive class and top 10 features of negative class for both feature sets **Set 1** and **Set 2** using values of `feature_log_prob_` parameter of [MultinomialNB](#) and print their corresponding feature names

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.

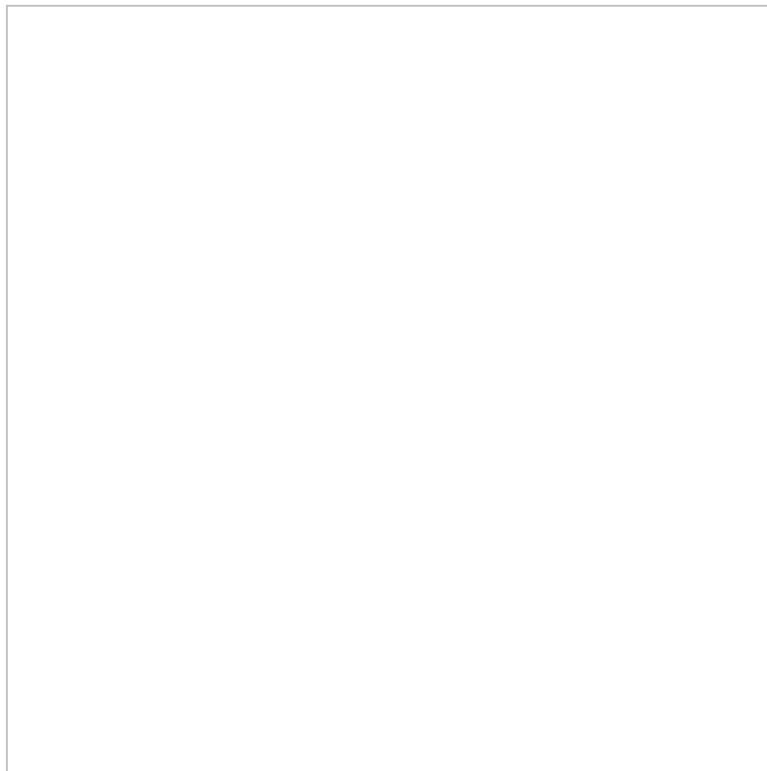
- 
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
- 

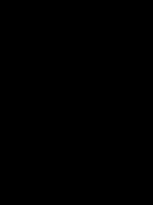
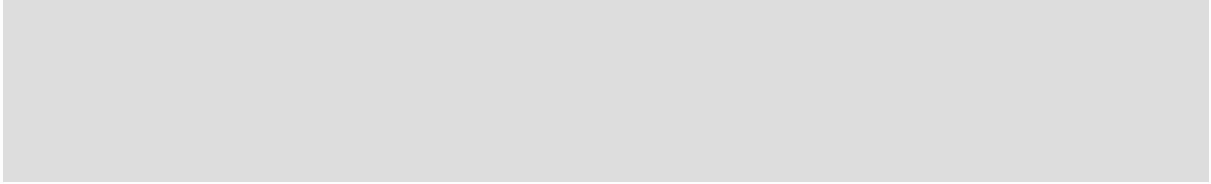
- Along with plotting ROC curve, you need to print the [confusion matrix](#) with predicted and original labels of test data points. Please visualize your confusion matrices using [seaborn heatmaps](#).



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](#)





Naive Bayes

Applying Naive Bayes on BOW, SET 1

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

In [69]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatenating a sparse
  matrix and a dense matrix :)
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_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_test.shape)
```

(73196, 16721)

(36052, 16721)

In [70]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB(class_prior=[0.5,0.5])

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

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

clf.fit(X_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 [71]:

```
#Output of GridSearchCV
print('Best score: ',clf.best_score_)
print('k value with best score: ',clf.best_params_)
print('='*75)
print('Train AUC scores')
print(clf.cv_results_['mean_train_score'])
print('CV AUC scores')
print(clf.cv_results_['mean_test_score'])
```

```
Best score:  0.707213591087
k value with best score:  {'alpha': 0.5}
=====
=====
Train AUC scores
[ 0.81779161  0.81657633  0.81590233  0.813888
52  0.81278816  0.80952747
```

```

0.80774387 0.80231383 0.79914251 0.787587
49 0.77886733 0.7378775
0.70903074 0.61428804 0.51378455 0.500029
02 0.5 0.5
0.50009023 0.50017585]
CV AUC scores
[ 0.68249025 0.68633061 0.68812029 0.692562
22 0.69453741 0.6991523
0.70106637 0.70500916 0.70632589 0.707213
59 0.70536326 0.68598149
0.66854748 0.59887913 0.51049098 0.500029
02 0.5 0.5
0.50009025 0.50016964]

```

In [72]:

```

def pred_prob(clf, data):
    y_pred = []
    y_pred = clf.predict_proba(data)[:,-1]
    return y_pred

```

In [73]:

```

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

```

Out[73]:

```

{'alpha': 0.5}

```

In [74]:

```

best_alpha1=clf.best_params_['alpha']

```

In [75]:

```

# https://scikit-learn.org/stable/modules/generated/sklearn.m

```

```
etrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

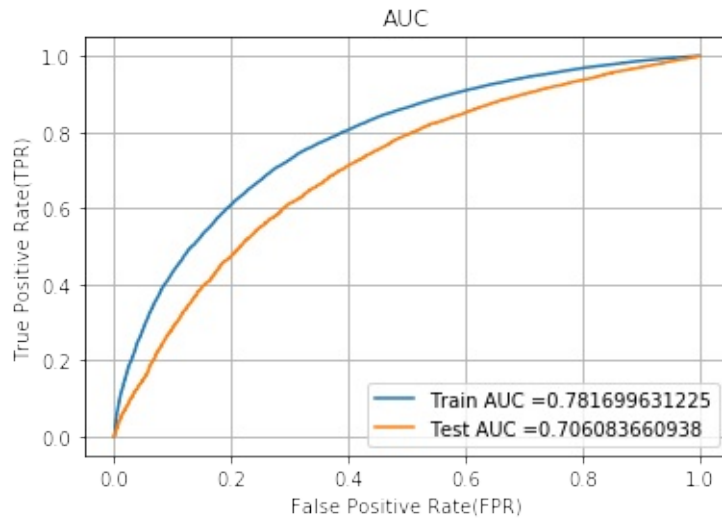
nb_bow = MultinomialNB(alpha = best_alpha1, class_prior = [0.5, 0.5])

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 = pred_prob(nb_bow, X_train)
y_test_pred = pred_prob(nb_bow, X_test)

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

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



In [76]:

```
# 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 [77]:

```
#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 $tpr \cdot (1 - fpr)$ 0.51113281892

3 for threshold 0.52

Train confusion matrix

```
[[ 8012  3071]
 [18196 43917]]
```

In [78]:

```
#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[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
f677a79940>
```



In [79]:

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

Test confusion matrix

```
[[ 3406  2053]
 [ 9495 21098]]
```

In [80]:

```
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[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff66d706a58>
```



Applying Naive Bayes on TFIDF, SET 2

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

In [81]:

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

X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_tfidf, train_title_tfidf, 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_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_tfidf, test_title_tfidf, 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_test.shape)
```

(73196, 16721)
(36052, 16721)

In [82]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB(class_prior=[0.5,0.5])

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

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

clf.fit(X_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 [83]:

```
#Output of GridSearchCV
print('Best score: ',clf.best_score_)
print('k value with best score: ',clf.best_params_)
print('='*75)
print('Train AUC scores')
print(clf.cv_results_['mean_train_score'])
print('CV AUC scores')
print(clf.cv_results_['mean_test_score'])
```

```
Best score:  0.673273602422
k value with best score:  {'alpha': 0.1}
=====
=====
Train AUC scores
```

```

[ 0.80974912  0.80813688  0.80727253  0.804752
08  0.80338999  0.79929022
    0.79692095  0.78840092  0.78202954  0.749774
11  0.72203169  0.63206791
    0.59768112  0.54932865  0.53898194  0.528409
37  0.52783355  0.52879129
    0.52983554  0.53066737]
CV AUC scores
[ 0.65822702  0.66092264  0.66215427  0.665136
08  0.66643788  0.66940298
    0.67060668  0.67296781  0.6732736   0.665270
33  0.65239194  0.59872102
    0.57609639  0.54224131  0.5345112   0.526778
48  0.52676911  0.52820972
    0.52942139  0.53037774]

```

In [84]:

```

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

```

Out[84]:

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

In [85]:

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

In [86]:

```

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

nb_tfidf = MultinomialNB(alpha = best_alpha2, class_prior = [0
.5, 0.5])

```

```

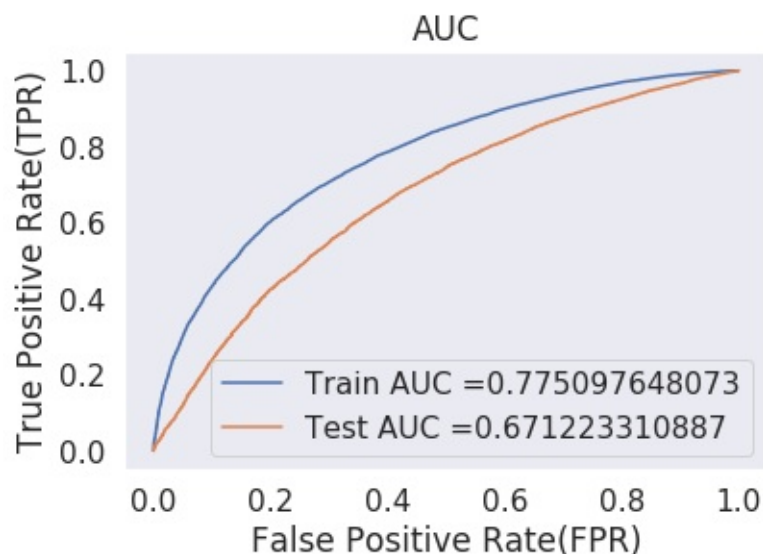
nb_tfidf.fit(X_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be
# probability estimates of the positive class
# not the predicted outputs

y_train_pred = pred_prob(nb_tfidf, X_train)
y_test_pred = pred_prob(nb_tfidf, X_test)

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

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

```



In [87]:

```
#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 $tpr \cdot (1 - fpr)$ 0.49729341387

7 for threshold 0.506

Train confusion matrix

```
[[ 7978  3105]  
 [19203 42910]]
```

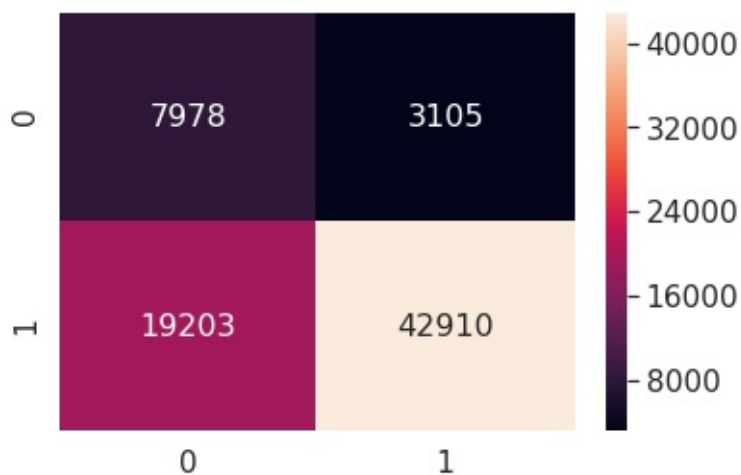
In [88]:

```
#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[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f  
f66e43b198>
```



In [89]:

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

Test confusion matrix

```
[[ 3207 2252]
 [10079 20514]]
```

In [90]:

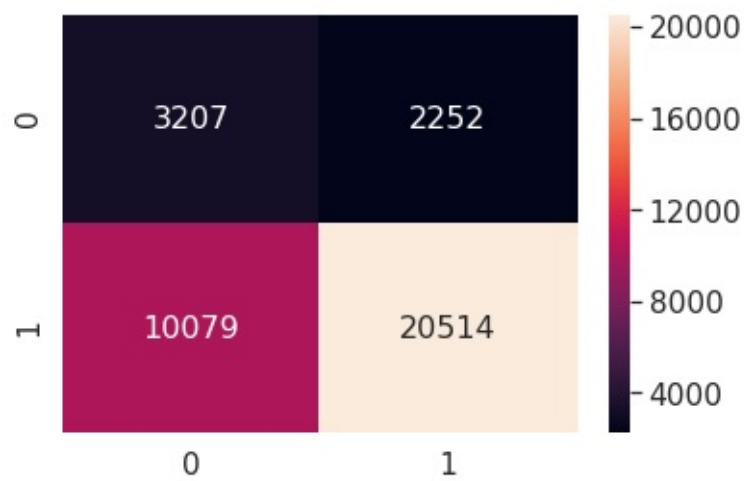
```
print("Test data confusion matrix")

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[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff66f438588>
```

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

In [91]:

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

```
# with the same hstack function we are concatenating a sparse matrix and a dense matrix :)
```

```
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_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_test.shape)
```

```
(73196, 16721)
```

```
(36052, 16721)
```

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

In [92]:

```
bow_features_names = []
```

In [93]:

```
for feature in vectorizer_cat.get_feature_names() :  
    bow_features_names.append(feature)  
  
for feature in vectorizer_subcat.get_feature_names() :  
    bow_features_names.append(feature)  
  
for feature in vectorizer_school_state.get_feature_names() :  
    bow_features_names.append(feature)  
  
for feature in vectorizer_grade.get_feature_names() :  
    bow_features_names.append(feature)  
  
for feature in vectorizer_prefix.get_feature_names() :  
    bow_features_names.append(feature)
```

In [94]:

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

In [95]:

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

In [96]:

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

In [97]:

```
len(bow_features_names)
```

Out[97]:

16721

In [101]:

```
#https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes  
#Feature importance in Naive Bayes  
  
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()  
    #class 0  
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()  
    #class1  
print('Top 10 features from negative class:')  
print(np.take(bow_features_names, neg_class_prob_sorted[:10])  
      )  
print('-'*50)  
print('Top 10 features from positive class:')  
print(np.take(bow_features_names, pos_class_prob_sorted[:10])  
      )
```

```
Top 10 features from negative class:  
['moon' 'investigation' 'uncomfortably' 'moments'  
 'jackets' 'calkins'  
  'backpack' 'ivy' 'hustle' 'grammatical']  
-----  
----  
Top 10 features from positive class:  
['junk' 'classical' 'ten' 'recite' 'tell' 'fingers'  
 'harness' 'discs'  
  'text' 'rack']
```

Top 10 important features of negative and positive class from

SET 2(TFIDF)

In [102]:

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

```
# with the same hstack function we are concatenating a sparse matrix and a dense matrix :)
```

```
X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_tfidf, train_title_tfidf, 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_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_tfidf, test_title_tfidf, 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_test.shape)
```

```
(73196, 16721)
```

```
(36052, 16721)
```

In [103]:

```
tfidf_features_names = []
```

In [104]:

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

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

```
tfidf_features_names.append(feature)

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

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

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

In [105]:

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

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

In [106]:

```
tfidf_features_names.append('price')
tfidf_features_names.append('teacher_number_of_previously_pos
ted_projects')
```

In [107]:

```
len(tfidf_features_names)
```

Out[107]:

16721

In [108]:

```
#https://stackoverflow.com/questions/50526898/how-to-get-feat  
ure-importance-in-naive-bayes  
#Feature importance in Naive Bayes
```

```

neg_class_prob_sorted = nb_tfidf.feature_log_prob_[0, :].argsort()
#class 0
pos_class_prob_sorted = nb_tfidf.feature_log_prob_[1, :].argsort()
#class1
print('Top 10 features from negative class:')
print(np.take(tfidf_features_names, neg_class_prob_sorted[:10]))
print('-'*50)
print('Top 10 features from positive class:')
print(np.take(tfidf_features_names, pos_class_prob_sorted[:10]))

```

Top 10 features from negative class:

```

['minds' 'conforming' 'myers' 'inquisitive' 'hello' 'heights' 'huddled'
 'hall' 'myplate' '2012']

```

```

-----
----

```

Top 10 features from positive class:

```

['exponential' 'dell' 'saying' 'phenomenon' 'livestock' 'baked' 'league'
 'addressing' 'lights' 'educate']

```

Conclusions

In [109]:

```
# 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 Parameter", "Test-AUC"]

x.add_row(["BOW", "Naive Bayes", 0.5, 0.70])
x.add_row(["TFIDF", "Naive Bayes", 0.1, 0.67])

print(x)
```

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