DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

- Literature & Writing, Social Sciences

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

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and

improve their school lives?"

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

In [5]:

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

In [6]:

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

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

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

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

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.1 Reading Data

In [7]:

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

In [8]:

```
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 [9]:

```
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[9]:

	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 [10]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & 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('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [11]:

```
sub_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"
```

```
Math&Science
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

1.3 Text preprocessing

In [12]:

```
# 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 [13]:

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

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

In [14]:

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

In [15]:

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

```
My students are English learners that are 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 s
chool. \r\n\r\n We have over 24 languages represented in our English Learner program with students at e
very level of mastery.  We also have over 40 countries represented with the families within our school.
very 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, bel
iefs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Ou
r 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 ba
rriers for parents to be able to help their child learn phonetics, letter recognition, and other readin
g skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of t
he English language even if no one at home is able to assist.  All families with students within the Le
vel 1 proficiency status, will be a offered to be a part of this program.  These educational videos wil
l be specially chosen by the English Learner Teacher and will be sent home regularly to watch.  The vid
eos 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 the
se videos and educational dvd's for the years to come for other EL students.\r\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. \r\nThe school has a vibrant community that loves to get togethe
r and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes tha
t students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, an
d games. At the end of the year the school hosts a carnival to celebrate the hard work put in during th
e school year, with a dunk tank being the most popular activity.My students will use these five brightl
y colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of
ten in the classroom and not enough for each student to have an individual one, they will be used in a
variety of ways. During independent reading time they will be used as special chairs students will each
use on occasion. I will utilize them in place of chairs at my small group tables during math and readin
g times. The rest of the day they will be used by the students who need the highest amount of movement
in their life in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is missing,
my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. Wh
en the students are sitting in group with me on the Hokki Stools, they are always moving, but at the sa
me 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. \r\n\r\nWe ask a lot of students to sit for 7 hou
rs 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 t
ake away the barrier that exists in schools for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desk
s, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to c
reate a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class i
s made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I
school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our s
chool is an \"open classroom\" concept, which is very unique as there are no walls separating the class
rooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all t
he information and experiences and keep on wanting more.With these resources such as the comfy red thro
w pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help creat
e the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom en
vironment is very important in the success in each and every child's education. The nautical photo prop
s will be used with each child as they step foot into our classroom for the first time on Meet the Teac
her evening. I'll take pictures of each child with them, have them developed, and then hung in our clas
sroom ready for their first day of 4th grade.  This kind gesture will set the tone before even the firs
t day of school! The nautical thank you cards will be used throughout the year by the students as they
create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make o
ur classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of m
y own pocket on resources to get our classroom ready. Please consider helping with this project to make
our new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students
. I teach in a Title I school where most of the students receive free or reduced price lunch.  Despite
their disabilities and limitations, my students love coming to school and come eager to learn and explo
re.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in
a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they
say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross
motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to s
it and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the ke
y to our success. The number toss and color and shape mats can make that happen. My students will forge
t they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great tea
cher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-Amer
ican, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2%
African-American students. Most of the students are on free or reduced lunch. We aren't receiving docto
rs, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspirin
g minds of young children and we focus not only on academics but one smart, effective, efficient, and d
isciplined students with good character.In our classroom we can utilize the Bluetooth for swift transit
ions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due t
ions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due t
o the volume of my speaker my students can't hear videos or books clearly and it isn't making the lesso
ns as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause
and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are n
eeded for the day and has an extra part to it I can use.  The table top chart has all of the letter, wo
rds and pictures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [16]:

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

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

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

In [17]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students
. I teach in a Title I school where most of the students receive free or reduced price lunch.  Despite
their disabilities and limitations, my students love coming to school and come eager to learn and explo
re.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in
a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they
say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross
motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to
sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the k
ey to our success. The number toss and color and shape mats can make that happen. My students will forg
et they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
```

In [18]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t working past their limitations.     The materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced price lunch.  Despite thei
r disabilities and limitations, my students love coming to school and come eager to learn and explore.H
ave you ever felt like you had ants in your pants and you needed to groove and move as you were in a me
eting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.
Wobble chairs are the answer and I love then because they develop their core, which enhances gross moto
r and in Turn fine motor skills.   They also want to learn through games, my kids do not want to sit an
d do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to
our success. The number toss and color and shape mats can make that happen. My students will forget the
y are doing work and just have the fun a 6 year old deserves.nannan
```

In [19]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitive del
ays gross fine motor delays to autism They are eager beavers and always strive to work their hardest wo
rking past their limitations The materials we have are the ones I seek out for my students I teach in a
rking past their limitations The materials we have are the ones I seek out for my students I teach in a
Title I school where most of the students receive free or reduced price lunch Despite their disabilitie
s and limitations my students love coming to school and come eager to learn and explore Have you ever f
elt 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 ar
e the answer and I love then because they develop their core which enhances gross motor and in Turn fin
e motor skills They also want to learn through games my kids do not want to sit and do worksheets They
want to learn to count by jumping and playing Physical engagement is the key to our success The number
toss and color and shape mats can make that happen My students will forget they are doing work and just
have the fun a 6 year old deserves nannan
```

In [20]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself'
, \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 't
heir',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these',
'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'd
o', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'whil
e', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'bef
ore', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'a
gain', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each
', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', '
m', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn
't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't",
'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't",
'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [21]:

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

```
100%|██████████| 109248/109248 [01:47<00:00, 1014.63it/s]
```

In [22]:

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

Out[22]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross fin
e motor delays autism they eager beavers always strive work hardest working past limitations the materi
als ones i seek students i teach title i school students receive free reduced price lunch despite disab
ilities limitations students love coming school come eager learn explore have ever felt like ants pants
needed groove move meeting this kids feel time the want able move learn say wobble chairs answer i love
develop core enhances gross motor turn fine motor skills they also want learn games kids not want sit w
orksheets they want learn count jumping playing physical engagement key success the number toss color s
hape mats make happen my students forget work fun 6 year old deserves nannan'
```

1.4 Preprocessing of `project_title`

1.4 Preprocessing of `project_title`

In [23]:

```
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [24]:

```
sent1 = decontracted(project_data['project_title'].values[2000])
print(sent1)
print("="*50)
```

```
Steady Stools for Active Learning
==================================================
```

In [25]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e.lower() not in stopwords)
    preprocessed_title.append(sent1.lower().strip())
```

```
100%|██████████| 109248/109248 [00:04<00:00, 24675.12it/s]
```

In [26]:

```
project_catogories = list(project_data['project_grade_category'].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

project_cat_list = []
for i in project_catogories:
    temp = ""
    for j in i.split(','):
        j = j.replace(' ','_') # we are placeing all the ' '(space)
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('-','_')
    project_cat_list.append(temp.strip())

project_data['clean_projectcategories'] = project_cat_list
project_data.drop(['project_grade_category'], 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_projectcategories'].values:
    my_counter.update(word.split())

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

In [27]:

```
project_data['clean_projectcategories']=project_data['clean_projectcategories'].str.lower()
```

In [28]:

```
#for teacher prefix
#https://www.geeksforgeeks.org/python-pandas-dataframe-fillna-to-replace-null-values-in-dataframe/
```

```
project_data["teacher_prefix"].fillna( method ='ffill', inplace = True)
```

In [29]:

```
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 [30]:

```
project_data['clean_essays'] = preprocessed_essays
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
```

1. count the total no of words in essay and make new feature column and add it to dataset
2. same for titles

In [31]:

```
X_essa=[]
for i in project_data['clean_essays']:
    b=len(i.split())
    X_essa.append(b)
project_data['no_essay']=X_essa
```

In [32]:

```
project_data['clean_titles'] = preprocessed_title
```

In [33]:

```
X_tri=[]
for i in project_data['clean_titles']:
    b=len(i.split())
    X_tri.append(b)
project_data['notitlewords']=X_tri
```

In [34]:

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

In [35]:

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

Out[35]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

In [36]:

```
project_data.count()
```

Out[36]:

```
Unnamed: 0                                      109248
id                                              109248
teacher_id                                      109248
teacher_prefix                                  109248
school_state                                    109248
project_submitted_datetime                      109248
project_resource_summary                        109248
teacher_number_of_previously_posted_projects    109248
project_is_approved                             109248
clean_categories                                109248
clean_subcategories                             109248
essay                                           109248
clean_projectcategories                         109248
price                                           109248
quantity                                        109248
clean_essays                                    109248
no_essay                                        109248
clean_titles                                    109248
notitlewords                                    109248
dtype: int64
```

2. Decision Tree

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

In [37]:

```python
y=project_data['project_is_approved'].values
project_data.drop(['project_is_approved'] , axis=1, inplace = True)
X=project_data
```

In [38]:

```python
X.head(2)
```

Out[38]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

SPLITTING USING TRAIN_TEST_SPLIT

In [39]:

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

In [40]:

```python
#Shape of training , test and cross validation data
```

```
print("X_train {0} ||  Y_train {1}".format(X_train.shape,y_train.shape))
print("X_cv {0}  || Y_cv {1}".format(X_cv.shape,y_cv.shape))
print("X_test {0} || Y_test {1}".format(X_test.shape,y_test.shape))
```

```
X_train (49041, 18) ||  Y_train (49041,)
X_cv (24155, 18)  || Y_cv (24155,)
X_test (36052, 18) || Y_test (36052,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 vectorizing categorical data

In [41]:

```
vectorizer_clean = CountVectorizer()
vectorizer_clean.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_ccat_ohe = vectorizer_clean.transform(X_train['clean_categories'].values)
X_cv_ccat_ohe = vectorizer_clean.transform(X_cv['clean_categories'].values)
X_test_ccat_ohe = vectorizer_clean.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_ccat_ohe.shape, y_train.shape)
print(X_cv_ccat_ohe.shape, y_cv.shape)
print(X_test_ccat_ohe.shape, y_test.shape)
print(vectorizer_clean.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 9) (49041,)
(24155, 9) (24155,)
(36052, 9) (36052,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_scienc
e', 'music_arts', 'specialneeds', 'warmth']
====================================================================================================
```

In [42]:

```
 vectorizer_clsub = CountVectorizer()
vectorizer_clsub.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cscat_ohe = vectorizer_clsub.transform(X_train['clean_subcategories'].values)
X_cv_cscat_ohe = vectorizer_clsub.transform(X_cv['clean_subcategories'].values)
X_test_cscat_ohe = vectorizer_clsub.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_cscat_ohe.shape, y_train.shape)
print(X_cv_cscat_ohe.shape, y_cv.shape)
print(X_test_cscat_ohe.shape, y_test.shape)
print(vectorizer_clsub.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 30) (49041,)
(24155, 30) (24155,)
(36052, 30) (36052,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_careerprep', 'co
mmunityservice', 'earlydevelopment', 'economics', 'environmentalscience', 'esl', 'extracurricular', 'fi
nancialliteracy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wellness', 'history_
geography', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutritioneducation', 'other', 'p
arentinvolvement', 'performingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'war
mth']
====================================================================================================
```

In [43]:

```
#FOR SCHOOL STATE
vectorizer_school = CountVectorizer()
vectorizer_school.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer_school.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer_school.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer_school.transform(X_test['school_state'].values)
```

```
print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer_school.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
['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']
====================================================================================================
```

In [44]:

```
vectorizer_cp = CountVectorizer()
vectorizer_cp.fit(X_train['clean_projectcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cpro_ohe = vectorizer_cp.transform(X_train['clean_projectcategories'].values)
X_cv_cpro_ohe = vectorizer_cp.transform(X_cv['clean_projectcategories'].values)
X_test_cpro_ohe = vectorizer_cp.transform(X_test['clean_projectcategories'].values)

print("After vectorizations")
print(X_train_cpro_ohe.shape, y_train.shape)
print(X_cv_cpro_ohe.shape, y_cv.shape)
print(X_test_cpro_ohe.shape, y_test.shape)
print(vectorizer_cp.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 4) (49041,)
(24155, 4) (24155,)
(36052, 4) (36052,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
====================================================================================================
```

In [45]:

```
vectorizer_teacher = CountVectorizer()
vectorizer_teacher.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer_teacher.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer_teacher.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer_teacher.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer_teacher.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 5) (49041,)
(24155, 5) (24155,)
(36052, 5) (36052,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
====================================================================================================
```

2.2.2 Vectorizing Numerical Features

PRICE

In [46]:

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

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

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

QUANTITY

In [47]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [48]:

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

X_train_quan_norm = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_quan_norm = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quan_norm = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_quan_norm.shape, y_train.shape)
print(X_cv_quan_norm.shape, y_cv.shape)
print(X_test_quan_norm.shape, y_test.shape)
print("="*100)
```

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

NO of previous posted project

In [49]:

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

X_train_tno_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.
reshape(-1,1))
X_cv_tno_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshap
e(-1,1))
X_test_tno_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.re
shape(-1,1))
```

```
print("After vectorizations")
print(X_train_tno_norm.shape, y_train.shape)
print(X_cv_tno_norm.shape, y_cv.shape)
print(X_test_tno_norm.shape, y_test.shape)
print("="*100)
```

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

No of words in titles

In [50]:

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

X_train_titleno_norm = normalizer.transform(X_train['notitlewords'].values.reshape(-1,1))
X_cv_titleno_norm = normalizer.transform(X_cv['notitlewords'].values.reshape(-1,1))
X_test_titleno_norm = normalizer.transform(X_test['notitlewords'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_titleno_norm.shape, y_train.shape)
print(X_cv_titleno_norm.shape, y_cv.shape)
print(X_test_titleno_norm.shape, y_test.shape)
print("="*100)
```

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

No of words in essay

In [51]:

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

X_train_essayno_norm = normalizer.transform(X_train['no_essay'].values.reshape(-1,1))
X_cv_essayno_norm = normalizer.transform(X_cv['no_essay'].values.reshape(-1,1))
X_test_essayno_norm = normalizer.transform(X_test['no_essay'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_essayno_norm.shape, y_train.shape)
print(X_cv_essayno_norm.shape, y_cv.shape)
print(X_test_essayno_norm.shape, y_test.shape)
print("="*100)
```

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

2.3 Make Data Model Ready: encoding eassay, and project_title

BAG OF WORDS

In [52]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizerb = CountVectorizer(min_df=10, max_features=5000)
vectorizerb.fit(X_train['clean_essays'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizerb.transform(X_train['clean_essays'].values)
X_cv_essay_bow = vectorizerb.transform(X_cv['clean_essays'].values)
X_test_essay_bow = vectorizerb.transform(X_test['clean_essays'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================
```

In [53]:

```
# BOW project titles
from sklearn.feature_extraction.text import CountVectorizer
vectorizert = CountVectorizer(min_df=10, max_features=5000)
vectorizert.fit(X_train['clean_titles'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_titles_bow = vectorizert.transform(X_train['clean_titles'].values)
X_cv_titles_bow = vectorizert.transform(X_cv['clean_titles'].values)
X_test_titles_bow = vectorizert.transform(X_test['clean_titles'].values)

print("After vectorizations")
print(X_train_titles_bow.shape, y_train.shape)
print(X_cv_titles_bow.shape, y_cv.shape)
print(X_test_titles_bow.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(49041, 2006) (49041,)
(24155, 2006) (24155,)
(36052, 2006) (36052,)
====================================================================================================
```

TFIDF

In [54]:

```
#FOR ESSAY
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tf = TfidfVectorizer(min_df=10)
vectorizer_tf.fit(X_train['clean_essays'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer_tf.transform(X_train['clean_essays'].values)
X_cv_essay_tfidf = vectorizer_tf.transform(X_cv['clean_essays'].values)
X_test_essay_tfidf = vectorizer_tf.transform(X_test['clean_essays'].values)

print(X_train_essay_tfidf.shape)
print(X_cv_essay_tfidf.shape)
print(X_test_essay_tfidf.shape)
```

```
(49041, 12136)
(24155, 12136)
(36052, 12136)
```

In [55]:

```
#for project title
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectKBest, chi2
```

```
vectorizer_t = TfidfVectorizer(min_df=10, max_features=5000)
vectorizer_t.fit(X_train['clean_titles'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_titles_tfidf = vectorizer_t.transform(X_train['clean_titles'].values)
X_cv_titles_tfidf = vectorizer_t.transform(X_cv['clean_titles'].values)
X_test_titles_tfidf = vectorizer_t.transform(X_test['clean_titles'].values)
print("Train shape:",X_train_titles_tfidf.shape)
print("CV shape:",X_cv_titles_tfidf.shape)
print("Test shape:",X_test_titles_tfidf.shape)
```

```
Train shape: (49041, 2006)
CV shape: (24155, 2006)
Test shape: (36052, 2006)
```

In [56]:

```
from tqdm import tqdm_notebook as tq
```

In [57]:

```
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tq(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')
```

```
Loading Glove Model

Done. 1917494  words loaded!
```

In [58]:

```
words_train_essays = []

for i in X_train['clean_essays']:
    words_train_essays.extend(i.split(' '))
```

In [59]:

```
## Find the total number of words in the Train data of Essays.

print("all the words in the corpus", len(words_train_essays))
```

```
all the words in the corpus 7425494
```

In [60]:

```
## Find the unique words in this set of words

words_train_essay = set(words_train_essays)
print("the unique words in the corpus", len(words_train_essay))
```

```
the unique words in the corpus 41355
```

In [61]:

```
## Find the words present in both Glove Vectors as well as our corpus.

inter_words = set(model.keys()).intersection(words_train_essay)

print("The number of words that are present in both glove vectors and our corpus are {} which \
is nearly {}% ".format(len(inter_words), np.round((float(len(inter_words))/len(words_train_essay))*100)
))
```

```
The number of words that are present in both glove vectors and our corpus are 37941 which is nearly 92.
0%
```

In [62]:

```
words_corpus_train_essay = {}
```

```
words_glove = set(model.keys())

for i in words_train_essay:
    if i in words_glove:
        words_corpus_train_essay[i] = model[i]

print("word 2 vec length", len(words_corpus_train_essay))
```

```
word 2 vec length 37941
```

In [63]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-an
d-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_corpus_train_essay, f)
```

In [64]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-an
d-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())
```

train essay for avg w2v

In [65]:

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

avg_w2v_vectors_train = [];

for sentence in tq(X_train['clean_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
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

```
49041
300
```

TEST TITLES

In [66]:

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

avg_w2v_vectors_test = [];

for sentence in tq(X_test['clean_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
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
36052
300
```

CROSS VALIDATION

In [67]:

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

avg_w2v_vectors_cv = [];

for sentence in tq(X_cv['clean_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
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))
```

```
24155
300
```

TRAIN TITLES

In [68]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tq(X_train['clean_titles']): # for each title
    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
    avg_w2v_vectors_titles_train.append(vector)

print(len(avg_w2v_vectors_titles_train))
print(len(avg_w2v_vectors_titles_train[0]))
```

```
49041
300
```

TEST TITLES

In [69]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tq(X_test['clean_titles']): # for each title
    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
    avg_w2v_vectors_titles_test.append(vector)

print(len(avg_w2v_vectors_titles_test))
print(len(avg_w2v_vectors_titles_test[0]))
```

```
36052
300
```

CROSS VALIDATION TITLES

In [70]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tq(X_cv['clean_titles']): # for each title
    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
    avg_w2v_vectors_titles_cv.append(vector)

print(len(avg_w2v_vectors_titles_cv))
print(len(avg_w2v_vectors_titles_cv[0]))
```

```
24155
300
```

TFIDF Weighted W2V

In [71]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['clean_essays'])
# 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 [72]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tq(X_train['clean_essays']): # 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
    tfidf_w2v_vectors_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))
```

```
49041
300
```

In [73]:

```
# compute average word2vec for each review.
#test essay

tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tq(X_test['clean_essays']): # 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
    tfidf_w2v_vectors_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
36052
300
```

In [74]:

```
# compute average word2vec for each review.
#cross validation essay
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tq(X_cv['clean_essays']): # 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
    tfidf_w2v_vectors_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

```
24155
300
```

TRAIN TITLES

In [75]:

```
tfidf_w2v_vectors_titles_train = [];

for sentence in tq(X_train['clean_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
    tfidf_w2v_vectors_titles_train.append(vector)

print(len(tfidf_w2v_vectors_titles_train))
print(len(tfidf_w2v_vectors_titles_train[0]))
```

```
49041
300
```

In [76]:

```
# compute average word2vec for each review.
#test titles
tfidf_w2v_vectors_titles_test = [];

for sentence in tq(X_test['clean_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
    tfidf_w2v_vectors_titles_test.append(vector)

print(len(tfidf_w2v_vectors_titles_test))
print(len(tfidf_w2v_vectors_titles_test[0]))
```

```
36052
300
```

In [77]:

```
# compute average word2vec for each review.
#cross validation titles
tfidf_w2v_vectors_titles_cv = [];

for sentence in tq(X_cv['clean_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
    tfidf_w2v_vectors_titles_cv.append(vector)

print(len(tfidf_w2v_vectors_titles_cv))
print(len(tfidf_w2v_vectors_titles_cv[0]))
```

```
24155
300
```

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

COMBINING ALL FEATURES

In [78]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_trs1 = hstack((X_train_ccat_ohe , X_train_cscat_ohe , X_train_state_ohe, X_train_cpro_ohe , X_train_teacher_ohe, X_train_essay_bow, X_train_titles_bow , X_train_price_norm, X_train_quan_norm , X_train_tno_norm)).tocsr()
X_cvs1 = hstack((X_cv_ccat_ohe , X_cv_cscat_ohe , X_cv_state_ohe, X_cv_cpro_ohe , X_cv_teacher_ohe, X_cv_essay_bow, X_cv_titles_bow , X_cv_price_norm, X_cv_quan_norm , X_cv_tno_norm)).tocsr()
X_tes1 = hstack((X_test_ccat_ohe , X_test_cscat_ohe , X_test_state_ohe, X_test_cpro_ohe , X_test_teacher_ohe, X_test_essay_bow, X_test_titles_bow , X_test_price_norm, X_test_quan_norm , X_test_tno_norm)).tocsr()
```

In [79]:

```
print("Final Data matrix")
print(X_trs1.shape, y_train.shape)
print(X_cvs1.shape, y_cv.shape)
print(X_tes1.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 7108) (49041,)
(24155, 7108) (24155,)
(36052, 7108) (36052,)
====================================================================================================
```

In [80]:

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

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

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

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

In [81]:

```
from sklearn.model_selection import GridSearchCV
from sklearn import tree
```

In [82]:

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

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

    return y_data_pred
```

as we have to train this model using two hyperparameter we use 3D plot to visualize best hyperparameters

In [83]:

```
from sklearn.metrics import roc_auc_score
dt_bow=tree.DecisionTreeClassifier()
train_auc=[]
cv_auc=[]
max_depth=[1,5, 10, 50, 100, 500]
min_samples_split=[5,10,50,100,500]
for i in tq(max_depth):
    for j in (min_samples_split):
        dt_bow=tree.DecisionTreeClassifier(max_depth=i,min_samples_split=j)
        dt_bow.fit(X_trs1,y_train)
        y_train_pred = batch_predict(dt_bow, X_trs1)
        y_cv_pred = batch_predict(dt_bow, X_cvs1)
        # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posit
ive class
        # not the predicted outputs
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

WE HAVE X-axis points from train AUC and CV-AUC

1. our y-axis will be either max_depth or min_samples_split

In [79]:

```
len(train_auc)
```

Out[79]:

```
30
```

In [80]:

```
len(cv_auc)
```

Out[80]:

```
30
```

In [81]:

```
cvx1=cv_auc
```

In [82]:

```
len(cvx1)
```

Out[82]:

```
30
```

In [83]:

```
cvy1= pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,5
0,100,500], index = cvx1)
```

In [84]:

```
cvz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = cvx1)
```

In [85]:

```
trx1=train_auc
```

In [86]:

```
try1=pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50
,100,500], index = trx1)
```

In [87]:

```
trz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = trx1)
```

In [88]:

```
%matplotlib inline
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [89]:

```
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''<script src="/static/components/requirejs/require.js"></script>'
''))
  init_notebook_mode(connected=False)
```

In [90]:

```
# https://plot.ly/python/3d-axes/
```

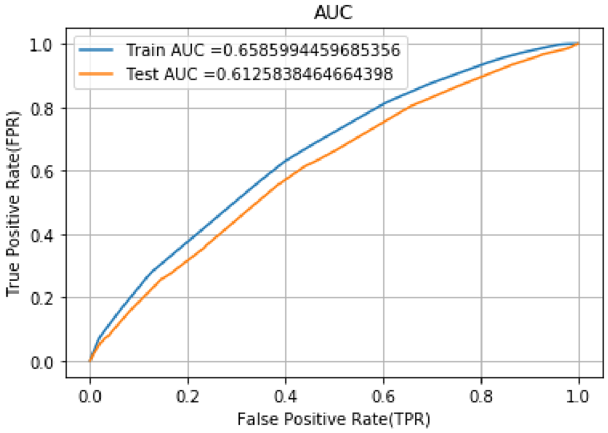
```
trace1 = go.Scatter3d(x=cvx1,y=cvy1,z=cvz1, name = 'Cross validation')
trace2 = go.Scatter3d(x=trx1,y=try1,z=trz1, name = 'train')
data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='AUC'),
        yaxis = dict(title='min_sample_splits'),
        zaxis = dict(title='max_depth'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
max_depth,min_samples_split=10,500
```

In [91]:

```
from sklearn.metrics import roc_curve, auc

model = tree.DecisionTreeClassifier(max_depth = 10, min_samples_split =500)

model.fit(X_trs1, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive clas
s
# not the predicted outputs

y_train_pred = batch_predict(model, X_trs1)
y_test_pred = batch_predict(model, X_tes1)

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(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



2. plot graph wiz

In [92]:

```
bow_names=[]
for a in vectorizer_clean.get_feature_names():
    bow_names.append(a)
```

In [93]:

```
for a in vectorizer_clsub.get_feature_names():
    bow_names.append(a)
```

In [94]:

```
for a in vectorizer_school.get_feature_names():
    bow_names.append(a)
```

In [95]:

```
for a in vectorizer_cp.get_feature_names():
    bow_names.append(a)
```

In [96]:

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

In [97]:

```
for a in vectorizerb.get_feature_names():
    bow_names.append(a)
```

In [98]:

```
for a in vectorizert.get_feature_names():
    bow_names.append(a)
```

In [99]:

```
bow_names.append('teacher_no_of_previously_posted_project')
bow_names.append('price')
bow_names.append('quantity')
```

In [100]:

```
len(bow_names)
```

Out[100]:

```
7068
```

THIS IS DONE TO GET ALL THE FEATURE NAME FROM COUNTVECTORIZER

In [101]:

```
X_trs1.shape
```

Out[101]:

```
(49041, 7068)
```

In [102]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(max_depth=3)
```

In [103]:

```
mo=model.fit(X_trs1,y_train)
```

AFTER FITTING THE DATA WE PLOT THE TREE USING GRAPH WIZ
https://medium.com/@rnbrown/creating-and-visualizing-decision-trees-with-python-f8e8fa394176

In [104]:

```
# Visualize data
import graphviz
from sklearn import tree
from graphviz import Source

dot_data = tree.export_graphviz(model,out_file=None ,feature_names=bow_names)
graph = graphviz.Source(dot_data)
graph.render("Bow tree1",view = True)
```

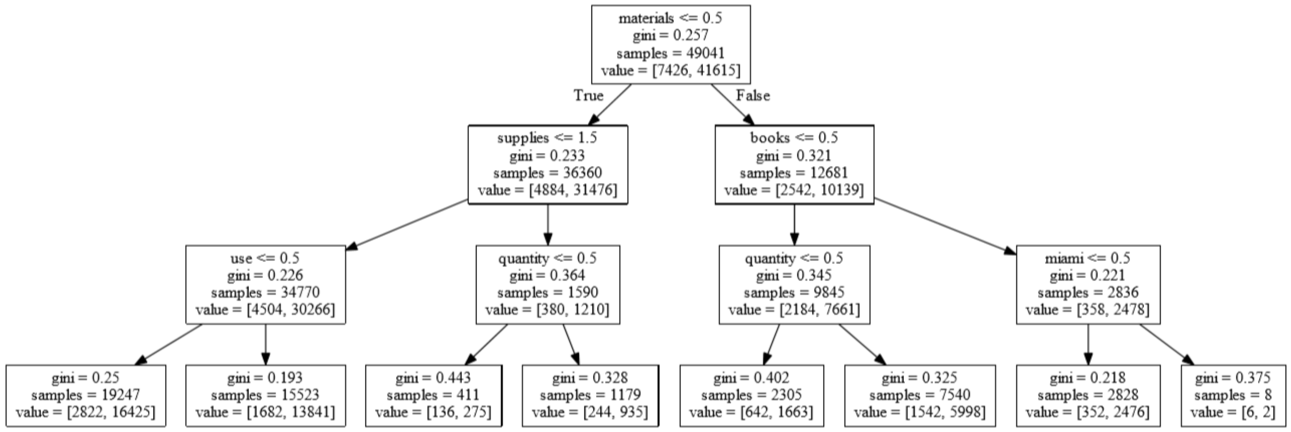
Out[104]:

```
'Bow tree1.pdf'
```

In [105]:

```
from IPython.display import Image
Image(filename='TREE.png')
```

Out[105]:



CONFUSION MATRIX

In [106]:

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

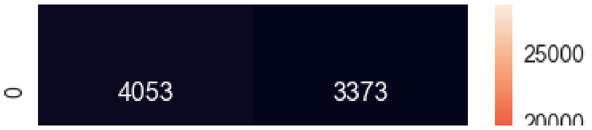
```
the maximum value of tpr*(1-fpr) 0.24790372649970413 for threshold 0.838
```

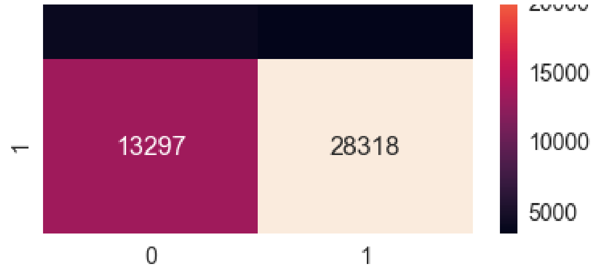
In [107]:

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

Out[107]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20120770a20>
```





In [108]:

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

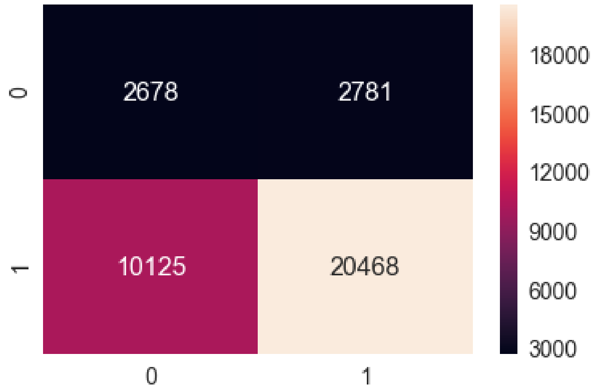
```
the maximum value of tpr*(1-fpr) 0.24991100035599859 for threshold 0.838
```

In [109]:

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

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x20120254ac8>
```



In [110]:

```
test=X_test_essay_bow.todense()
test.shape
```

Out[110]:

```
(36052, 5000)
```

In [111]:

```
bow_features= vectorizerb.get_feature_names()
```

we get the feature names of bag of words of essays

In [112]:

```
bow_features[0:5]
```

Out[112]:

```
['00', '000', '10', '100', '1000']
```

In [113]:

```
y_test_con=list(y_test)
```

In [114]:

```
y_test_con[0:5]
```

Out[114]:

```
[1, 1, 1, 1, 1]
```

__false positive will store index of the element whose actual value is 0 but predicted as 1
the index will be of features whose threshold of test pred is less than max threshold and y[i] = 0

the index will be of features whose threshold of test pred is less than max threshold and y[i] = 0

In [115]:

```
false_positive=[]
fp_count=0
for i in range(len(y_test_pred)):
    if(y_test_pred[i]<=0.842 and y_test_con[i]==0):
        false_positive.append(i)
        fp_count=fp_count+1
    else:
        continue
```

In [116]:

```
false_positive[0:5]
```

Out[116]:

```
[9, 10, 41, 51, 78]
```

In [117]:

```
fp_count
```

Out[117]:

```
2941
```

In [118]:

```
df2=pd.DataFrame(test)
```

In [119]:

```
dffina=df2.iloc[false_positive,:]
```

In [120]:

```
dffina.head(2)
```

Out[120]:

	0	1	2	3	4	5	6	7	8	9	...	4990	4991	4992	4993	4994	4995	4996	4997	4998	4999
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

2 rows × 5000 columns

In [121]:

```
dffina.shape
```

Out[121]:

```
(2941, 5000)
```

In [122]:

```
main_features=[]
sum1=0
for i in range(5000):
    sum1=dffina[i].sum()
    if sum1>=20:
        main_features.append(i)
    else:
        continue
```

In [123]:

```
len(main_features)
```

Out[123]:

```
2205
```

In [124]:

In [124]:

```
feature_name=[]
for i in (main_features):
    feature_name.append(bow_features[i])
```

select some feature to show in word cloud as more no of word will be messy
so we select some best words
https://www.geeksforgeeks.org/generating-word-cloud-python/

In [125]:

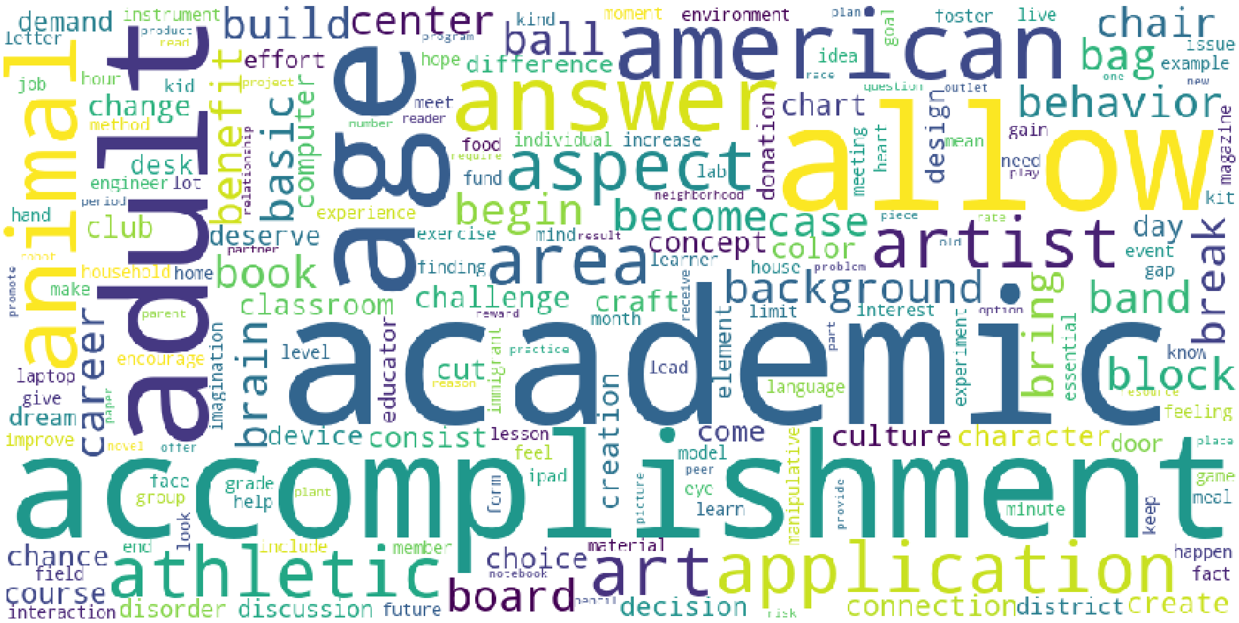
```
feature_name[0:5]
```

Out[125]:

```
['000', '10', '100', '11', '12']
```

In [126]:

```
from wordcloud import WordCloud
unique_string=(" ").join(feature_name)
wordcloud = WordCloud(width = 1000, height = 500, background_color ='white').generate(unique_string)
plt.figure(figsize=(15,10))
plt.imshow(wordcloud)
plt.axis("off")
plt.savefig("Word_Cloud_tfidf"+".png", bbox_inches='tight')
plt.show()
plt.close()
```



BOX PLOT BETWEEN FALSE POSITIVE AND PRICE

In [127]:

```
df_price=pd.DataFrame(X_test['price'])
```

In [128]:

```
df_boxi=df_price.iloc[false_positive,:]
```

In [129]:

```
df_boxi.head(2)
```

Out[129]:

	price
21229	546.10

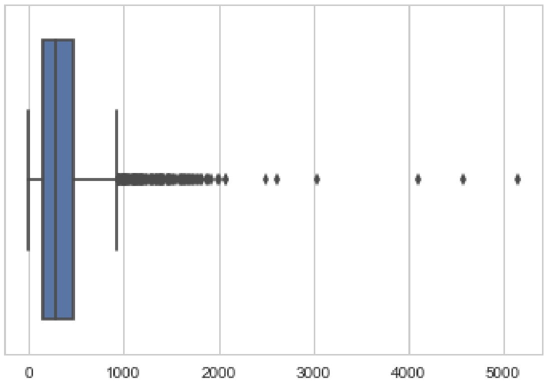
33852	35.74 **price**

In [130]:

```
import seaborn as sns
sns.set(style="whitegrid")
sns.boxplot(df_boxi.values)
```

Out[130]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2011f9b7da0>
```



OBSERVATION
from above graph it is clearly seen that most of the project that were rejected but but our model predicted as positive costs less than 500 dollars

PDF WITH TEACHER NO OF PREVIOUSLY POSTED PROJECT

In [131]:

```
df_teacher=pd.DataFrame(X_test['teacher_number_of_previously_posted_projects'])
```

In [132]:

```
df_pdf=df_teacher.iloc[false_positive,:]
```

In [133]:

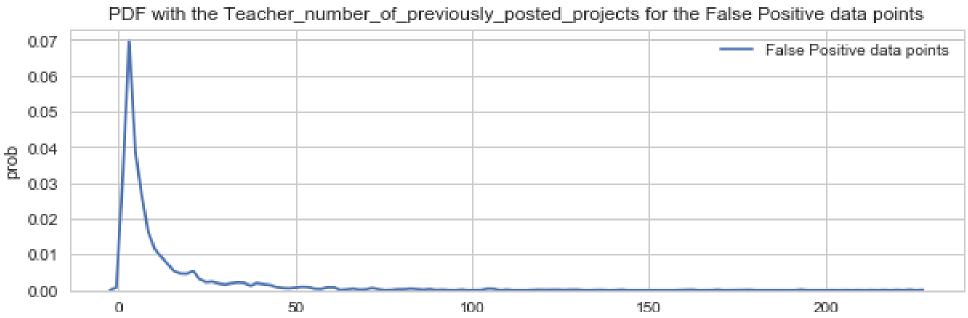
```
df_pdf.head(2)
```

Out[133]:

	teacher_number_of_previously_posted_projects
21229	5
33852	1

In [134]:

```
plt.figure(figsize=(10,3))
sns.distplot(df_pdf, hist=False, label="False Positive data points")
plt.title('PDF with the Teacher_number_of_previously_posted_projects for the False Positive data points
')
plt.xlabel('Teacher_number_of_previously_posted_projects')
plt.ylabel('prob')
plt.legend()
plt.show()
```



OBSERVATION

Majority of the teacher no of peviously projected project is near to zero almost 10%

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

In [84]:

```
from sklearn.model_selection import GridSearchCV
from sklearn import tree
```

In [85]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_tfidf = hstack((X_train_ccat_ohe , X_train_cscat_ohe , X_train_state_ohe, X_train_cpro_ohe , X_tra
in_teacher_ohe, X_train_essay_tfidf, X_train_titles_tfidf , X_train_price_norm, X_train_quan_norm , X_t
rain_tno_norm)).tocsr()
X_cr_tfidf = hstack((X_cv_ccat_ohe , X_cv_cscat_ohe , X_cv_state_ohe, X_cv_cpro_ohe , X_cv_teacher_ohe,
X_cv_essay_tfidf, X_cv_titles_tfidf , X_cv_price_norm, X_cv_quan_norm , X_cv_tno_norm)).tocsr()
X_te_tfidf = hstack((X_test_ccat_ohe , X_test_cscat_ohe , X_test_state_ohe, X_test_cpro_ohe , X_test_te
acher_ohe, X_test_essay_tfidf, X_test_titles_tfidf , X_test_price_norm, X_test_quan_norm , X_test_tno_n
orm)).tocsr()
```

HYPERPARAMETER : MAX_DEPTH and MIN_SAMPLES_SPLIT

In [86]:

```
from sklearn.metrics import roc_auc_score
dt_tfidf=tree.DecisionTreeClassifier()
train_auc=[]
cv_auc=[]
max_depth=[1,5, 10, 50, 100, 500]
min_samples_split=[5,10,50,100,500]
for i in tq(max_depth):
    for j in (min_samples_split):
        dt_tfidf=tree.DecisionTreeClassifier(max_depth=i,min_samples_split=j)
        dt_tfidf.fit(X_tr_tfidf,y_train)
        y_train_pred = batch_predict(dt_tfidf, X_tr_tfidf)
        y_cv_pred = batch_predict(dt_tfidf, X_cr_tfidf)
        # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posit
ive class
        # not the predicted outputs
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

In [138]:

```
len(train_auc)
```

Out[138]:

```
30
```

In [139]:

```
len(cv_auc)
```

Out[139]:

```
30
```

In [140]:

```
cvx1=cv_auc
```

In [141]:

```
len(cvx1)
```

Out[141]:

```
30
```

In [142]:

```
cvy1= pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,5
0,100,500], index = cvx1)
```

In [143]:

```
cvz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = cvx1)
```

In [144]:

```
trx1=train_auc
```

In [145]:

```
try1=pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50
,100,500], index = trx1)
```

In [146]:

```
trz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = trx1)
```

In [147]:

```
%matplotlib inline
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [148]:

```
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''<script src="/static/components/requirejs/require.js"></script>'
''))
  init_notebook_mode(connected=False)
```

In [149]:

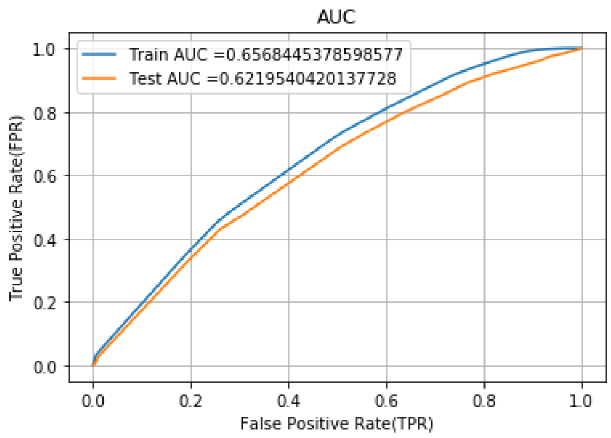
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=cvx1,y=cvy1,z=cvz1, name = 'Cross validation')
trace2 = go.Scatter3d(x=trx1,y=try1,z=trz1, name = 'train')
data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='AUC'),
        yaxis = dict(title='min_sample_split'),
        zaxis = dict(title='max_depth'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
max_depth,min_samples_split=10,100
```

In [87]:

```
from sklearn.metrics import roc_curve, auc

model = tree.DecisionTreeClassifier(max_depth = 10, min_samples_split =100)

model.fit(X_tr_tfidf, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive clas
s
# not the predicted outputs

y_train_pred = batch_predict(model, X_tr_tfidf)
y_test_pred = batch_predict(model, X_te_tfidf)

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(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [151]:

```
tfidf_names=[]
for a in vectorizer_clean.get_feature_names():
    tfidf_names.append(a)
```

In [152]:

```
for a in vectorizer_clsub.get_feature_names():
    tfidf_names.append(a)
```

In [153]:

```
for a in vectorizer_school.get_feature_names():
    tfidf_names.append(a)
```

In [154]:

```
for a in vectorizer_cp.get_feature_names():
    tfidf_names.append(a)
```

In [155]:

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

In [156]:

```
for a in vectorizer_tf.get_feature_names():
    tfidf_names.append(a)
```

In [157]:

```
for a in vectorizer_t.get_feature_names():
    tfidf_names.append(a)
```

In [158]:

```
tfidf_names.append('teacher_no_of_previously_posted_project')
tfidf_names.append('price')
tfidf_names.append('quantity')
```

In [159]:

```
len(tfidf_names)
```

Out[159]:

```
14261
```

In [160]:

```
X_te_tfidf.shape
```

Out[160]:

```
(36052, 14261)
```

In [161]:

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(max_depth=3)
```

In [162]:

```
mo=model.fit(X_tr_tfidf,y_train)
```

In [163]:

```
# Visualize data
import graphviz
from sklearn import tree
from graphviz import Source

dot_data = tree.export_graphviz(model,out_file=None ,feature_names=tfidf_names)
graph = graphviz.Source(dot_data)
graph.render("tfidftree",view = True)
```

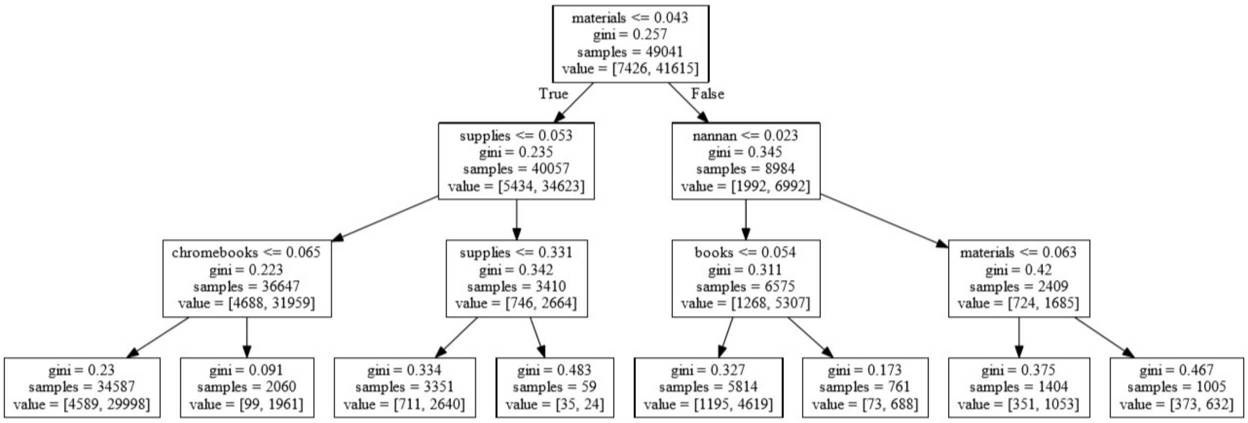
Out[163]:

```
'tfidftree.pdf'
```

In [164]:

```
from IPython.display import Image
Image(filename='tfidftree.jpg')
```

Out[164]:



CONFUSION MATRIX

In [165]:

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

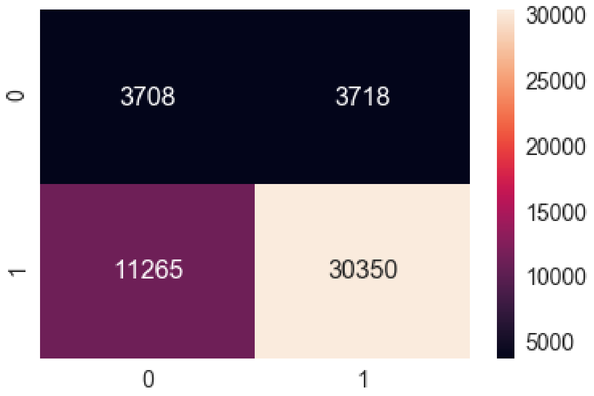
```
the maximum value of tpr*(1-fpr) 0.24999954665365479 for threshold 0.846
```

In [166]:

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

Out[166]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2011fa6a278>
```



In [167]:

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

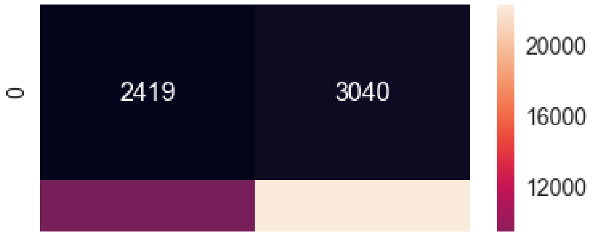
```
the maximum value of tpr*(1-fpr) 0.24804292224060248 for threshold 0.846
```

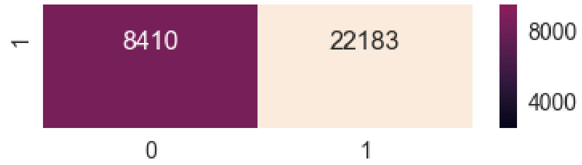
In [168]:

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

Out[168]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2011faab128>
```





In [169]:

```
test=X_test_essay_tfidf.todense()
test.shape
```

Out[169]:

```
(36052, 12193)
```

In [170]:

```
tfidf_features= vectorizer_tf.get_feature_names()
```

In [171]:

```
tfidf_features[0:5]
```

Out[171]:

```
['00', '000', '10', '100', '1000']
```

In [172]:

```
y_test_con=list(y_test)
```

In [173]:

```
y_test_con[0:5]
```

Out[173]:

```
[1, 1, 1, 1, 1]
```

In [174]:

```
false_positive=[]
fp_count=0
for i in range(len(y_test_pred)):
    if(y_test_pred[i]<=0.856 and y_test_con[i]==0):
        false_positive.append(i)
        fp_count=fp_count+1
    else:
        continue
```

In [175]:

```
false_positive[0:5]
```

Out[175]:

```
[9, 16, 41, 51, 78]
```

In [176]:

```
fp_count
```

Out[176]:

```
2488
```

In [177]:

```
df2=pd.DataFrame(test)
```

In [178]:

```
dffina=df2.iloc[false_positive,:]
```

In [179]:

```
dffina.head(5)
```

Out[179]:

	0	1	2	3	4	5	6	7	8	9	...	12183	12184	12185	12186	12187	12188	12189	12190	12191	12192
9	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
16	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
41	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
51	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
78	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

5 rows × 12193 columns

In [180]:

```
dffina.shape
```

Out[180]:

```
(2488, 12193)
```

In [181]:

```
main_features=[]
sum1=0
for i in range(12134):
    sum1=dffina[i].sum()
    if sum1>=20:
        main_features.append(i)
    else:
        continue
```

In [182]:

```
len(main_features)
```

Out[182]:

```
187
```

In [183]:

```
feature_name=[]
for i in (main_features):
    feature_name.append(tfidf_features[i])
```

In [184]:

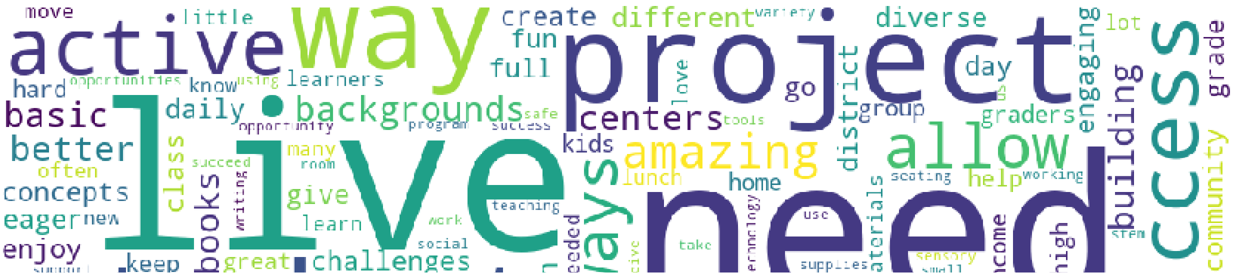
```
feature_name[0:5]
```

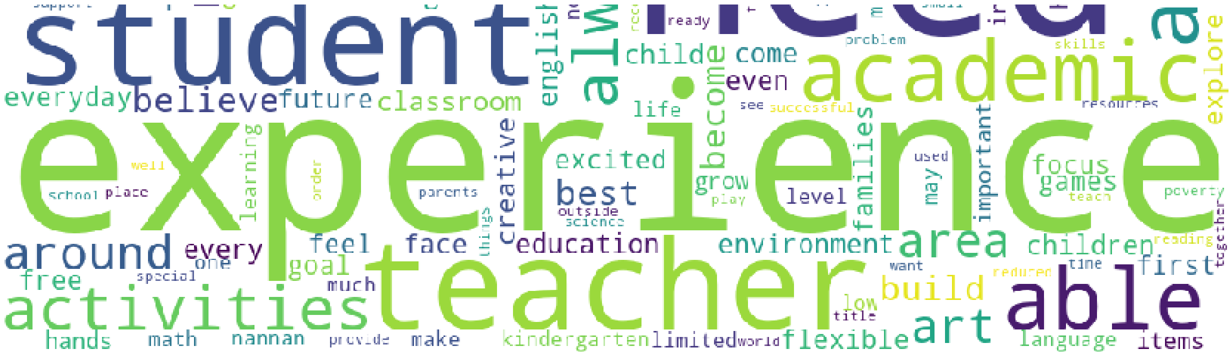
Out[184]:

```
['able', 'academic', 'access', 'active', 'activities']
```

In [185]:

```
from wordcloud import WordCloud
unique_string=(" ").join(feature_name)
wordcloud = WordCloud(width = 1000, height = 500, background_color ='white').generate(unique_string)
plt.figure(figsize=(15,10))
plt.imshow(wordcloud)
plt.axis("off")
plt.savefig("Word_Cloud_tfidf"+".png", bbox_inches='tight')
plt.show()
plt.close()
```





BOX PLOT BETWEEN FALSE POSITIVE AND PRICE

In [186]:

```
df_price=pd.DataFrame(X_test['price'])
```

In [187]:

```
df_boxi=df_price.iloc[false_positive,:]
```

In [188]:

```
df_boxi.head(2)
```

Out[188]:

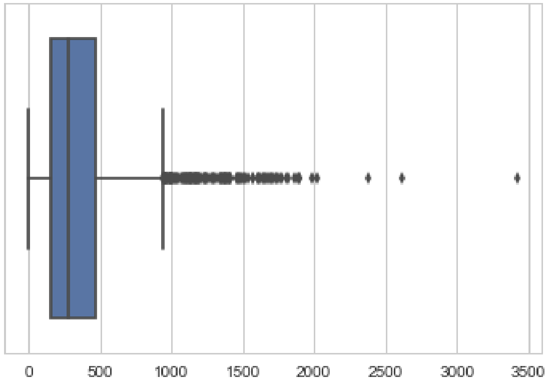
	price
21229	546.10
30510	989.25

In [189]:

```
import seaborn as sns
sns.set(style="whitegrid")
sns.boxplot(df_boxi.values)
```

Out[189]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2011d644438>
```



PDF WITH TEACHER NO OF PREVIOUSLY POSTED PROJECT

In [190]:

```
df_teacher=pd.DataFrame(X_test['teacher_number_of_previously_posted_projects'])
```

In [191]:

```
df_pdf=df_teacher.iloc[false_positive,:]
```

In [192]:

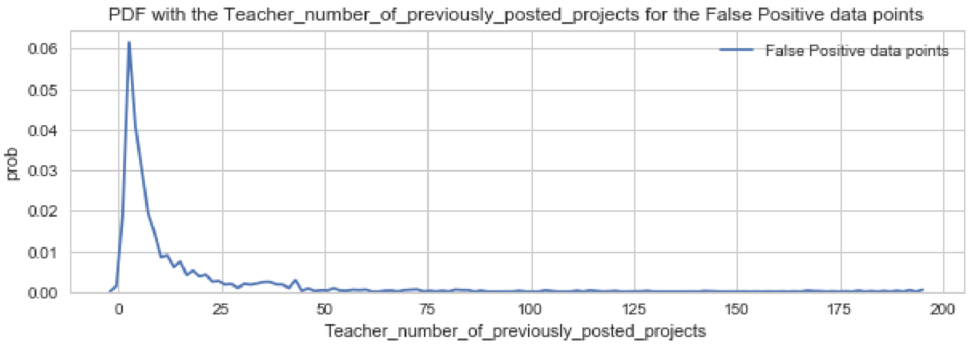
```
df_pdf.head(2)
```

Out[192]:

	teacher_number_of_previously_posted_projects
21229	5
30510	2

In [193]:

```
plt.figure(figsize=(10,3))
sns.distplot(df_pdf, hist=False, label="False Positive data points")
plt.title('PDF with the Teacher_number_of_previously_posted_projects for the False Positive data points
')
plt.xlabel('Teacher_number_of_previously_posted_projects')
plt.ylabel('prob')
plt.legend()
plt.show()
```



OBSERVATION

1.Box plot and pdf nearly gives same observation for both the model

1. most of the project that were rejected but but our model predicted as positive costs less than 500 dollars.
2. teacher no of previously posted projects is less than 10% for false positive

Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)

In [194]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_avg = hstack((X_train_ccat_ohe , X_train_cscat_ohe , X_train_state_ohe, X_train_cpro_ohe , X_train
_teacher_ohe,avg_w2v_vectors_train,avg_w2v_vectors_titles_train, X_train_price_norm, X_train_quan_norm
, X_train_tno_norm)).tocsr()
X_cr_avg = hstack((X_cv_ccat_ohe , X_cv_cscat_ohe , X_cv_state_ohe, X_cv_cpro_ohe , X_cv_teacher_ohe, a
vg_w2v_vectors_cv,avg_w2v_vectors_titles_cv , X_cv_price_norm, X_cv_quan_norm , X_cv_tno_norm)).tocsr()
X_te_avg = hstack((X_test_ccat_ohe , X_test_cscat_ohe , X_test_state_ohe, X_test_cpro_ohe , X_test_teac
her_ohe, avg_w2v_vectors_test,avg_w2v_vectors_titles_test, X_test_price_norm, X_test_quan_norm , X_test
_tno_norm)).tocsr()
```

In [86]:

```
from sklearn.metrics import roc_auc_score
dt_avg=tree.DecisionTreeClassifier()
train_auc=[]
cv_auc=[]
max_depth=[1,5, 10, 50, 100, 500]
min_samples_split=[5,10,50,100,500]
for i in tq(max_depth):
    for j in (min_samples_split):
        dt_avg=tree.DecisionTreeClassifier(max_depth=i,min_samples_split=j)
        dt_avg.fit(X_tr_avg,y_train)
        y_train_pred = batch_predict(dt_avg, X_tr_avg)
        y_cv_pred = batch_predict(dt_avg, X_cr_avg)
        # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posit
ive class
        # not the predicted outputs
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

In [196]:

```
len(train_auc)
```

Out[196]:

```
30
```

In [197]:

```
len(cv_auc)
```

Out[197]:

```
30
```

In [198]:

```
cvx1=cv_auc
```

In [199]:

```
len(cvx1)
```

Out[199]:

```
30
```

In [200]:

```
cvy1= pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,5
0,100,500], index = cvx1)
```

In [201]:

```
cvz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = cvx1)
```

In [202]:

```
trx1=train_auc
```

In [203]:

```
try1=pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50
,100,500], index = trx1)
```

In [204]:

```
trz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = trx1)
```

In [205]:

```
%matplotlib inline
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [206]:

```
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''<script src="/static/components/requirejs/require.js"></script>'
''))
  init_notebook_mode(connected=False)
```

In [207]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=cvx1,y=cvy1,z=cvz1, name = 'Cross validation')
trace2 = go.Scatter3d(x=trx1,y=try1,z=trz1, name = 'train')
```

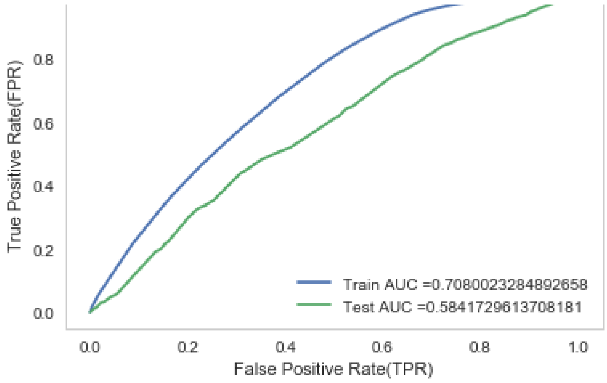
```
trace2 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'train')
data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='AUC'),
        yaxis = dict(title='min_sample_split'),
        zaxis = dict(title='max_depth'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
max_depth,min_samples_split=5,500
```

In [208]:

```
from sklearn.metrics import roc_curve, auc

model = tree.DecisionTreeClassifier(max_depth = 5, min_samples_split =500)

model.fit(X_tr_avg, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive clas
s
# not the predicted outputs

y_train_pred = batch_predict(model, X_tr_avg)
y_test_pred = batch_predict(model, X_te_avg)

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(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

In [209]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_weigh = hstack((X_train_ccat_ohe , X_train_cscat_ohe , X_train_state_ohe, X_train_cpro_ohe , X_train_teacher_ohe,tfidf_w2v_vectors_train,tfidf_w2v_vectors_titles_train, X_train_price_norm, X_train_quan_norm , X_train_tno_norm)).tocsr()
X_cr_weigh = hstack((X_cv_ccat_ohe , X_cv_cscat_ohe , X_cv_state_ohe, X_cv_cpro_ohe , X_cv_teacher_ohe, tfidf_w2v_vectors_cv,tfidf_w2v_vectors_titles_cv , X_cv_price_norm, X_cv_quan_norm , X_cv_tno_norm)).tocsr()
X_te_weigh = hstack((X_test_ccat_ohe , X_test_cscat_ohe , X_test_state_ohe, X_test_cpro_ohe , X_test_teacher_ohe, tfidf_w2v_vectors_test,tfidf_w2v_vectors_titles_test, X_test_price_norm, X_test_quan_norm , X_test_tno_norm)).tocsr()
```

In [86]:

```
from sklearn.metrics import roc_auc_score
dt_weigh=tree.DecisionTreeClassifier()
train_auc=[]
cv_auc=[]
max_depth=[1,5, 10, 50, 100, 500]
min_samples_split=[5,10,50,100,500]
for i in tq(max_depth):
    for j in (min_samples_split):
        dt_weigh=tree.DecisionTreeClassifier(max_depth=i,min_samples_split=j)
        dt_weigh.fit(X_tr_weigh,y_train)
        y_train_pred = batch_predict(dt_weigh, X_tr_weigh)
        y_cv_pred = batch_predict(dt_weigh, X_cr_weigh)
        # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
        # not the predicted outputs
        train_auc.append(roc_auc_score(y_train,y_train_pred))
        cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

In [211]:

```
len(train_auc)
```

Out[211]:

```
30
```

In [212]:

```
len(cv_auc)
```

Out[212]:

```
30
```

In [213]:

```
cvx1=cv_auc
```

In [214]:

```
len(cvx1)
```

Out[214]:

```
30
```

In [215]:

```
cvy1= pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,5
0,100,500], index = cvx1)
```

In [216]:

```
cvz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = cvx1)
```

In [217]:

```
trx1=train_auc
```

In [218]:

```
try1=pd.Series([5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50
,100,500], index = trx1)
```

In [219]:

```
trz1=pd.Series([1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,50,50,50,50,50,100,100,100,100,100,500,500,500,500,5
00],index = trx1)
```

In [220]:

```
%matplotlib inline
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np
```

In [221]:

```
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''<script src="/static/components/requirejs/require.js"></script>'
''))
  init_notebook_mode(connected=False)
```

In [222]:

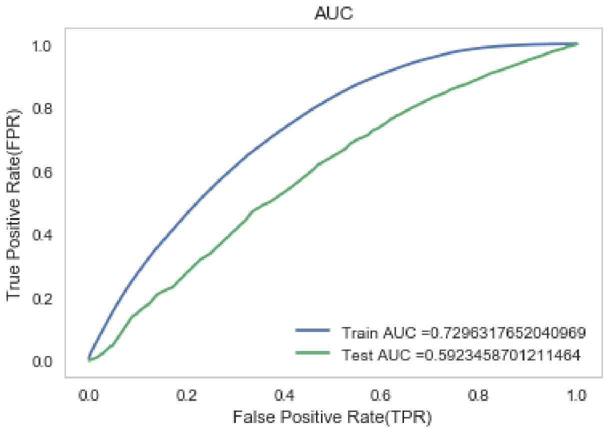
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=cvx1,y=cvy1,z=cvz1, name = 'Cross validation')
trace2 = go.Scatter3d(x=trx1,y=try1,z=trz1, name = 'train')
data = [trace1, trace2]
enable_plotly_in_cell()

layout = go.Layout(scene = dict(
        xaxis = dict(title='AUC'),
        yaxis = dict(title='min_sample_split'),
        zaxis = dict(title='max_depth'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [ ]:

```
max_depth,min_samples_split=5,100
```

In [223]:

```
from sklearn.metrics import roc_curve, auc

model = tree.DecisionTreeClassifier(max_depth = 5, min_samples_split =100)

model.fit(X_tr_weigh, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive clas
s
# not the predicted outputs

y_train_pred = batch_predict(model, X_tr_weigh)
y_test_pred = batch_predict(model, X_te_weigh)

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(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



SELECT 5k best feature from SET2(TFIDF):

In [88]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_tfidf_feat = hstack((X_train_ccat_ohe , X_train_cscat_ohe , X_train_state_ohe, X_train_cpro_ohe ,
X_train_teacher_ohe, X_train_essay_tfidf, X_train_titles_tfidf , X_train_price_norm, X_train_quan_norm
, X_train_tpe_norm)).tocsr()
, X_train_tno_norm)).tocsr()
X_cr_tfidf_feat= hstack((X_cv_ccat_ohe , X_cv_cscat_ohe , X_cv_state_ohe, X_cv_cpro_ohe , X_cv_teacher_
ohe, X_cv_essay_tfidf, X_cv_titles_tfidf , X_cv_price_norm, X_cv_quan_norm , X_cv_tno_norm)).tocsr()
X_te_tfidf_feat = hstack((X_test_ccat_ohe , X_test_cscat_ohe , X_test_state_ohe, X_test_cpro_ohe , X_te
st_teacher_ohe, X_test_essay_tfidf, X_test_titles_tfidf , X_test_price_norm, X_test_quan_norm , X_test_
tno_norm)).tocsr()
```

In [101]:

```
X_tr_tfidf_feat.shape
```

Out[101]:

```
(49041, 14244)
```

In [90]:

```
dt_feature=tree.DecisionTreeClassifier()
```

In [91]:

```
dt_feature.fit(X_tr_tfidf_feat,y_train)
```

Out[91]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
```

In [102]:

```
#https://stackoverflow.com/questions/49170296/scikit-learn-feature-importance-calculation-in-decision-t
rees
a=dt_feature.tree_.compute_feature_importances(normalize=False)
```

In [107]:

```
a=list(a)
```

In [108]:

```
index=[]
for i in range(14244):
    if a[i]>0:
        index.append(i)
```

In [109]:

```
len(index)
```

Out[109]:

```
2687
```

we get 2687 as important feature all other features are of 0 importance

In [111]:

```
data1=X_tr_tfidf_feat.todense()
```

In [113]:

```
X_train_feat=pd.DataFrame(data1)
```

In [117]:

```
X_train_feat=X_train_feat.iloc[:,index]
```

In [120]:

```
X_train_feat.head(2)
```

Out[120]:

	0	4	5	7	18	22	23	24	25	26	...	14189	14190	14217	14218	14222	14223	14227	14228	14234	14243
0	0.0	1.0	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	1.0
1	0.0	1.0	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	1.0

2 rows × 2687 columns

for Test

In [124]:

```
b=X_te_tfidf_feat.todense()
```

In [125]:

```
X_test_feat=pd.DataFrame(b)
```

In [126]:

```
X_test_feat=X_test_feat.iloc[:,index]
```

FOR crossValidation

In [129]:

```
c=X_cr_tfidf_feat.todense()
```

In [130]:

```
X_cr_feat=pd.DataFrame(c)
```

In [131]:

```
X_cr_feat=X_cr_feat.iloc[:,index]
```

SVM ON BEST FEATURES

In [132]:

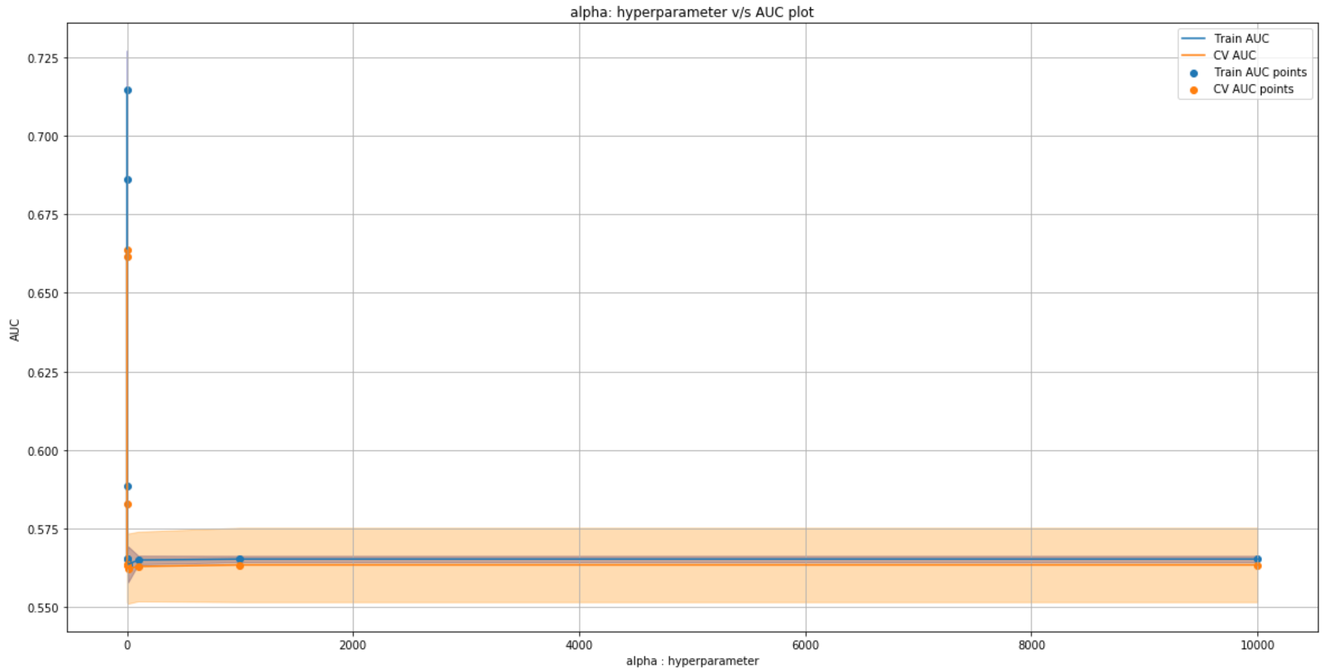
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
```

In [133]:

```
sv = SGDClassifier(loss='hinge', penalty='l2',class_weight='balanced')

parameters = {'alpha':[10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**1, 10**2, 10**3, 10**4]}

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

clf.fit(X_train_feat, 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 [134]:

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

plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.
3,color='darkblue')

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

plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')


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

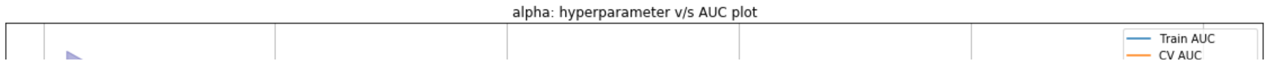


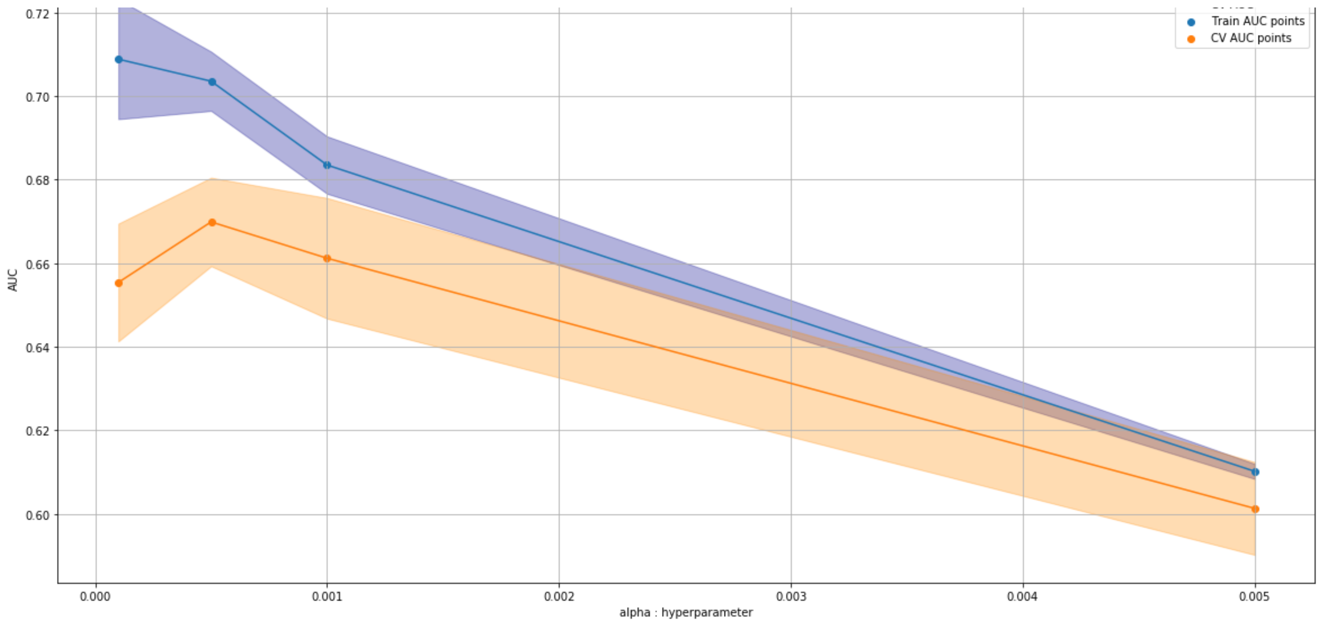
In [137]:

```
sv = SGDClassifier(loss='hinge', penalty='l2',class_weight='balanced')

parameters = {'alpha':[0.0001,0.0005,0.001,0.005]}

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

clf.fit(X_train_feat, 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 [138]:

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

plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.
3,color='darkblue')

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

plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')


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



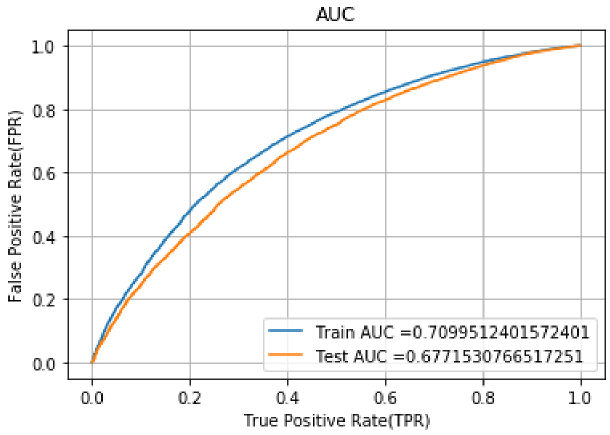


In [140]:

```
best_alpha=0.0005
```

In [141]:

```
from sklearn.metrics import roc_curve, auc


model = SGDClassifier(loss='hinge', penalty='l2', alpha=best_alpha,class_weight='balanced')

model.fit(X_train_feat, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive clas
s
# not the predicted outputs

y_train_pred = model.decision_function(X_train_feat)
y_test_pred = model.decision_function(X_test_feat)

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

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



In [142]:

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

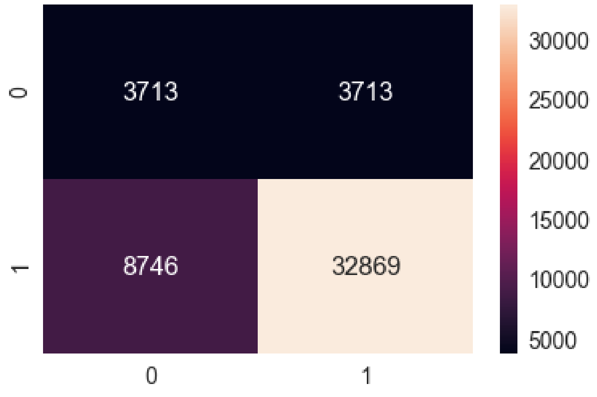
```
the maximum value of tpr*(1-fpr) 0.25 for threshold -0.559
```

In [143]:

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

Out[143]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2511d398e80>
```



In [144]:

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

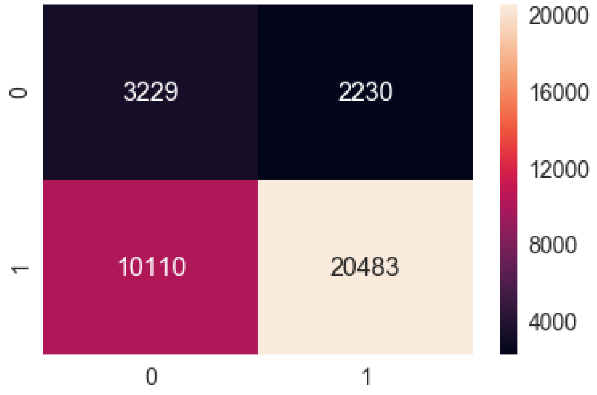
```
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold -0.341
```

In [145]:

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

Out[145]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2511d31abe0>
```



CONCLUSION

In [147]:

```
# 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", "HYPER PARAMETERS", "AUC"]

x.add_row(["BOW", "DECISION TREE", "max depth:10 min_samples_split:500 ", "0.61"])
x.add_row(["TFIDF", "DECISION TREE", "max depth:10 min_samples_split:100 ", "0.62"])
x.add_row(["AVG W2V", "DECISION TREE","max depth:5 min_samples_split: 500", "0.58"])
x.add_row(["TFIDF W2V", "DECISION TREE","max depth:5 min_samples_split:100 ", "0.59"])
x.add_row(["BEST_FEATURES", "SUPPORT VECTOR MACHINE","@lpha:0.005", "0.67"])
```

```
print(x)
```

```
+---------------+------------------------+-------------------------------------+------+
|   Vectorizer  |         Model          |           HYPER PARAMETERS          | AUC  |
+---------------+------------------------+-------------------------------------+------+
|      BOW      |     DECISION TREE      | max depth:10 min_samples_split:500  | 0.61 |
|     TFIDF     |     DECISION TREE      | max depth:10 min_samples_split:100  | 0.62 |
|    AVG W2V    |     DECISION TREE      |  max depth:5 min_samples_split: 500 | 0.58 |
|   TFIDF W2V   |     DECISION TREE      |  max depth:5 min_samples_split:100  | 0.59 |
| BEST_FEATURES | SUPPORT VECTOR MACHINE |             @lpha:0.005             | 0.67 |
+---------------+------------------------+-------------------------------------+------+
```

OBSERVATION

1. Highest AUC score which is obtained is 0.62 for Decision tree.
2. model is good because for every vectorizer AUC score is greater than 0.5
3. We have seen other model which gives more AUC score than Decision Tree
4. Training Time is high for Decision Tree.
5. Main advantage is Decision Tree is highly interpretable we get main features using graphwiz

END