DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*

Feature	Description
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

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

In [3]:

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

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

Out[4]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00

	id	description	quantity	price
1	p069069	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [5]:

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

1.2 preprocessing of `project_subject_categories`

In [6]:

```
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 & Hunge
r"]
        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 r
emoving '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 [7]:

```
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 & Hunge
r"]
        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 r
emoving 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>
"Math&Science"
```

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

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

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

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

1.3 Text preprocessing

In [8]:

```
# 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 [9]:

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

Out[9]:

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

In [10]:

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

In [11]:

```
# 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. 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
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 [12]:

```
# 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 [13]:

```
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 [14]:

```
# \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 [15]:

```
#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
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 [16]:

```
# 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 [17]:

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

```
100%|██████████| 109248/109248 [00:47<00:
00, 2320.27it/s]
```

In [18]:

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

Out[18]:

```
'kindergarten students varied disabilities ranging speech language delays cognitive delays gross fine m
otor delays autism eager beavers always strive work hardest working past limitations materials ones see
k students teach title school students receive free reduced price lunch despite disabilities limitation
s students love coming school come eager learn explore ever felt like ants pants needed groove move mee
ting kids feel time want able move learn say wobble chairs answer love develop core enhances gross moto
r turn fine motor skills also want learn games kids not want sit worksheets want learn count jumping pl
aying physical engagement key success number toss color shape mats make happen students forget work fun
6 year old deserves nannan'
```

In [19]:

```
# Updating dataframe for clean project title and remove old project title
project_data['clean_essay'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
project_data.head(2)
```

Out[19]:

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

1.4 Preprocessing of `project_title`

In [20]:

```
# similarly you can preprocess the titles also
# 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):
    sent = sentance.lower()
    sent = decontracted(sent)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_title.append(sent.strip())
```

```
100%|██████████| 109248/109248 [00:02<00:0
0, 52759.87it/s]
```

In [21]:

```
# after preprocesing
preprocessed_title[20000]
```

Out[21]:

```
'need move input'
```

In [22]:

```
# Updating dataframe for clean project title and remove old project title
project_data['clean_project_title'] = preprocessed_title
project_data.drop(['project_title'], axis=1, inplace=True)
project_data.head(2)
```

Out[22]:

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

Preprocessing project_grade

In [23]:

```
# similarly you can preprocess the project_grade also
# Combining all the above stundents
from tqdm import tqdm
preprocessed_grade = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_grade_category'].values):
    sent = sentance.lower()
    sent = decontracted(sent)
    sent = sent.replace(' ', '_')
    sent = sent.replace('-', '_')
    # https://gist.github.com/sebleier/554280
#     sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_grade.append(sent.strip())
```

```
100%|██████████| 109248/109248 [00:00<00:00
, 133875.71it/s]
```

In [24]:

```
preprocessed_grade[:10]
```

Out[24]:

```
['grades_prek_2',
 'grades_6_8',
 'grades_6_8',
 'grades_prek_2',
 'grades_prek_2',
 'grades_3_5',
 'grades_6_8',
 'grades_3_5',
 'grades_prek_2',
 'grades_prek_2']
```

In [25]:

```
# Updating dataframe for clean project title and remove old project title
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data['project_grade_category'] = preprocessed_grade
project_data.head(2)
```

Out[25]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay_1
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	My students are English learners that are work...
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Our students arrive to our school eager to lea...

In [26]:

```
# remove unnecessary column: https://cmdlinetips.com/2018/04/how-to-drop-one-or-more-columns-in-pandas-dataframe/
project_data = project_data.drop(['Unnamed: 0','id','teacher_id','project_submitted_datetime', \
                                  'project_essay_1','project_essay_2','project_essay_3','project_essay_4', \
                                  'project_resource_summary'], axis=1)
```

In [27]:

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

Out[27]:

	teacher_prefix	school_state	teacher_number_of_previously_posted_projects	project_is_approved	price	quantity	clean_categories
0	Mrs.	IN	0	0	154.6	23	Literacy_Language
1	Mr.	FL	7	1	299.0	1	History_Civics Health_Sports

Check whether each column contain NaN or Not

In [28]:

```
project_data['teacher_prefix'].isnull().values.any()
```

Out[28]:

```
True
```

In [29]:

```
project_data['school_state'].isnull().values.any()
```

Out[29]:

```
False
```

In [30]:

```
project_data['teacher_number_of_previously_posted_projects'].isnull().values.any()
```

Out[30]:

False

In [31]:

```
project_data['project_is_approved'].isnull().values.any()
```

Out[31]:

False

In [32]:

```
project_data['price'].isnull().values.any()
```

Out[32]:

False

In [33]:

```
project_data['quantity'].isnull().values.any()
```

Out[33]:

False

In [34]:

```
project_data['clean_categories'].isnull().values.any()
```

Out[34]:

False

In [35]:

```
project_data['clean_subcategories'].isnull().values.any()
```

Out[35]:

False

In [36]:

```
project_data['clean_essay'].isnull().values.any()
```

Out[36]:

False

In [37]:

```
project_data['clean_project_title'].isnull().values.any()
```

Out[37]:

False

In [38]:

```
project_data['project_grade_category'].isnull().values.any()
```

Out[38]:

False

Since we got 'teacher prefix' attributes which contain NaN. Let check how many NaN are contain in this attributes

In [39]:

```
project_data['teacher_prefix'].isnull().sum().sum()
```

Out[39]:

3

1.5 Preparing data for models

In [40]:

```
project_data.columns
```

Out[40]:

```
Index(['teacher_prefix', 'school_state',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'clean_essay', 'clean_project_title', 'project_grade_category'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [0]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
```

```
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sp
orts', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [0]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_
Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'Perf
ormingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geogr
aphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualA
rts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Liter
acy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [0]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [0]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

In [0]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

In [0]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [0]:

```
'''
```

```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(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')

# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


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

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


'''
```

Out[0]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words that are present in both glove vectors and our coupus",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n\'\'\'\n'
```

In [0]:

In [0]:

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

In [0]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|██████████| 109248/109248 [00:31<00:00, 3508.17it/s]
```

```
109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [0]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_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 [0]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_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.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|██████████| 109248/109248 [03:28<00
```

:00, 524.21it/s]

```
109248
300
```

In [0]:

```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

In [0]:

```
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 [0]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73 5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

In [0]:

```
price_standardized
```

Out[0]:

```
array([[0.00098843, 0.00191166, 0.00330448, ..., 0.00153418, 0.00046704,
        0.00070265]])
```

1.5.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [0]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [0]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[0]:

```
(109248, 16663)
```

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Computing Sentiment Scores

In [0]:

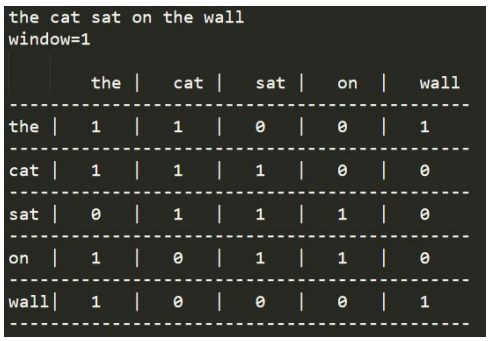
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students with t
he biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelligence
s i use a wide range\
of techniques to help all my students succeed students in my class come from a variety of different bac
kgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a caring com
munity of successful \
learners which can be seen through collaborative student project based learning in and out of the class
room kindergarteners \
in my class love to work with hands on materials and have many different opportunities to practice a sk
ill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the kinderg
arten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role play in
our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food i wil
l take their idea \
and create common core cooking lessons where we learn important math and writing concepts while cooking
delicious healthy \
food for snack time my students will have a grounded appreciation for the work that went into making th
e food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project would exp
and our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce
make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create our own cookbooks
to be printed and \
shared with families students will gain math and literature skills as well as a life long enjoyment for
healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975
```

```
D:\installed\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:20: UserWarning:

The twython library has not been installed. Some functionality from the twitter package will not be ava
ilable.
```

```
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

Assignment 11: TruncatedSVD

- step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title and then find the top 2k words) based on their `idf_` values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref)



- step 3 Use TruncatedSVD on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method

> - The shape of the matrix after TruncatedSVD will be 2000*n, i.e. each row represents a vector form of the corresponding word.
> - Vectorize the essay text and project titles using these word vectors. (while vectorizing, do ignore all the words which are not in top 2k words)

- step 4 Concatenate these truncatedSVD matrix, with the matrix with features
  - **school_state** : categorical data
  - **clean_categories** : categorical data
  - **clean_subcategories** : categorical data
  - **project_grade_category** :categorical data
  - **teacher_prefix** : categorical data
  - **quantity** : numerical data
  - **teacher_number_of_previously_posted_projects** : numerical data
  - **price** : numerical data
  - **sentiment score's of each of the essay** : numerical data
  - **number of words in the title** : numerical data
  - **number of words in the combine essays** : numerical data
  - **word vectors calculated in** step 3 : numerical data
- step 5: Apply GBDT on matrix that was formed in step 4 of this assignment, **DO REFER THIS BLOG: XGBOOST DMATRIX**
- **step 6:Hyper parameter tuning (Consider any two hyper parameters)**
  - **Find the best hyper parameter which will give the maximum AUC value**
  - **Find the best hyper paramter using k-fold cross validation or simple cross validation data**
  - **Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning**

In [0]:

```
import sys
import math

import numpy as np
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import roc_auc_score

# you might need to install this one
import xgboost as xgb
```

```
class XGBoostClassifier():
    def __init__(self, num_boost_round=10, **params):
        self.clf = None
        self.num_boost_round = num_boost_round
        self.params = params
        self.params.update({'objective': 'multi:softprob'})

    def fit(self, X, y, num_boost_round=None):
        num_boost_round = num_boost_round or self.num_boost_round
        self.label2num = {label: i for i, label in enumerate(sorted(set(y)))}
        dtrain = xgb.DMatrix(X, label=[self.label2num[label] for label in y])
        self.clf = xgb.train(params=self.params, dtrain=dtrain, num_boost_round=num_boost_round, verbos
e_eval=1)

    def predict(self, X):
        num2label = {i: label for label, i in self.label2num.items()}
        Y = self.predict_proba(X)
        y = np.argmax(Y, axis=1)
        return np.array([num2label[i] for i in y])

    def predict_proba(self, X):
        dtest = xgb.DMatrix(X)
        return self.clf.predict(dtest)

    def score(self, X, y):
        Y = self.predict_proba(X)[:,1]
        return roc_auc_score(y, Y)

    def get_params(self, deep=True):
        return self.params

    def set_params(self, **params):
        if 'num_boost_round' in params:
            self.num_boost_round = params.pop('num_boost_round')
        if 'objective' in params:
            del params['objective']
        self.params.update(params)
        return self


clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4,)
########################################################################
#               Change from here                                       #
########################################################################
parameters = {
    'num_boost_round': [100, 250, 500],
    'eta': [0.05, 0.1, 0.3],
    'max_depth': [6, 9, 12],
    'subsample': [0.9, 1.0],
    'colsample_bytree': [0.9, 1.0],
}

clf = GridSearchCV(clf, parameters)
X = np.array([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])
Y = np.array([0, 1, 0, 1, 0, 1])
clf.fit(X, Y)

# print(clf.grid_scores_)
best_parameters, score, _ = max(clf.grid_scores_, key=lambda x: x[1])
print('score:', score)
for param_name in sorted(best_parameters.keys()):
    print("%s: %r" % (param_name, best_parameters[param_name]))
```

```
score: 0.8333333333333334
colsample_bytree: 0.9
eta: 0.05
max_depth: 6
num_boost_round: 100
subsample: 0.9
```

2. TruncatedSVD

In [41]:

```
# Combine the train.csv and resource.csv
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
# https://www.geeksforgeeks.org/python-pandas-dataframe-sample/
# Take 50k dataset
project_data = project_data.sample(n=50000)
# Remove that row which contain NaN. We observed that only 3 rows that contain NaN
project_data = project_data[pd.notnull(project_data['teacher_prefix'])]
project_data.shape
```

Out[41]:

```
(49998, 11)
```

In [357]:

```
from sklearn.model_selection import train_test_split

tr,ts = train_test_split(project_data,test_size=0.2,random_state=1,
                                    stratify=project_data['project_grade_category'].values)
```

In [358]:

```
tr.shape, ts.shape
```

Out[358]:

```
((39998, 11), (10000, 11))
```

In [359]:

```
tr,cv_ = train_test_split(tr,test_size=0.25,random_state=1,
                                    stratify=tr['project_grade_category'].values)
```

In [360]:

```
tr.shape,cv_.shape
```

Out[360]:

```
((29998, 11), (10000, 11))
```

2.1 Selecting top 2000 words from `essay` and `project_title`

In [42]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [221]:

```
essay_title = tr['clean_essay'].map(str) + tr['clean_project_title'].map(str)
essay_title = pd.DataFrame(essay_title)
essay_title.reset_index(inplace=True, drop=True)
essay_title.head()
# pd.DataFrame(pd.concat([project_data['clean_essay'],project_data['clean_project_title']],ignore_index
```

```
=True))
```

Out[221]:

	0
0	many students come families struggle provide b...
1	students come assortment backgrounds many limi...
2	dance room filled inspirational quotes picture...
3	second grade class bubbling excitement adventu...
4	class kindergarten students come high poverty ...

In [222]:

```
essay_title.shape
```

Out[222]:

```
(29998, 1)
```

In [223]:

```
e_t = []
for i in range(len(essay_title)):
#     print(essay_title.iloc[i,0])
    e_t.append(str(essay_title.iloc[i,0]))

e_t[:2]
```

Out[223]:

```
['many students come families struggle provide basic necessities life already learned look bright side
try make best situation flexible resilient curious loving joy teach many struggle reading basic mathema
tics skills come school every day ready willing best work learn although often see bad things happen ne
ighborhood not lost hope things get better families providing motivating learning materials relevant li
ves encourage students dream big imagine reach stars older mobile classroom stark industrial looking fl
oor not best shape would like cheerful place students gather morning class meetings option sit carpet l
earning groups collaborate games projects colorful comfortable carpet classroom brighten learning envir
onment encourage collaboration provide seating options students reliable pencil sharpener pockets keep
sharpened pencils allow students maximize learning time math class nannanhome sweet home',
 'students come assortment backgrounds many limited exposure books experiences outside school books rea
d provided school library class text students special despite obstacles strong desire explore learn new
things mission keep passion learning strong creating memorable learning experiences always appreciative
creative learning events students like collaborate share ideas makes learning exciting fun table studen
ts express share learning feel students engaged table stools give students chance write articulate thou
ghts without using pencil paper also allows classroom chance use less paper school started recycling pr
ogram last year creating newer ways not use paper teaches students another way friendly planet definite
win win nannangoing greener']
```

In [239]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(e_t)
```

Out[239]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=1.0, max_features=None,
                min_df=10, ngram_range=(1, 1), norm='l2', preprocessor=None,
                smooth_idf=True, stop_words=None, strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)
```

In [240]:

```
# Number of features
```

```
# Number of features
len(vectorizer.get_feature_names())
```

Out[240]:

```
10516
```

In [241]:

```
# Display some idf value
vectorizer.idf_
```

Out[241]:

```
array([7.1657846 , 5.8117511 , 8.669862  , ..., 7.37709369, 8.05082279,
       8.91102405])
```

In [242]:

```
# Sort in descending order and take 2k top values and display idf values
index_sort = np.flip(np.argsort(vectorizer.idf_))[:2000]
vectorizer.idf_[index_sort]
```

Out[242]:

```
array([8.91102405, 8.91102405, 8.91102405, ..., 8.5363306 , 8.5363306 ,
       8.5363306 ])
```

In [243]:

```
# Store the top 2k feature names
essay_title_features = []
for i in index_sort:
    essay_title_features.append(vectorizer.get_feature_names()[i])

# Print top 10 feature names
print(essay_title_features[:5])
```

```
['zumba', 'amplified', 'nannanencouraging', 'nannaneverything', 'ambiance']
```

2.2 Computing Co-occurance matrix

In [49]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [2]:

```
def performrowupdation(sample_feature,A,index_value,window_size,N):
    # Initialize zeros as row vector feature
    m = np.zeros([1,N])

    # If the index value is in first position
    if index_value == 0:
        win_count = 0
        for i in range(1,len(A)):
```

```
            if win_count < window_size:
                if A[i] in sample_feature:
                    for j in range(len(sample_feature)):
                        if A[i] == sample_feature[j]:
                            m[0][j] += 1
            win_count += 1

    # If the index value is in last position
    elif index_value == (len(A)-1):
        win_count = 0
        for i in range(index_value-1,-1,-1):
            if win_count < window_size:
                if A[i] in sample_feature:
                    for j in range(len(sample_feature)):
                        if A[i] == sample_feature[j]:
                            m[0][j] += 1
            win_count += 1

    # If the position is in middle position
    else:
        # Window size in backward diretion
        win_count = 0
        for i in range(index_value-1,-1,-1):
            if win_count < window_size and i > -1:
                if A[i] in sample_feature:
                    for j in range(len(sample_feature)):
                        if A[i] == sample_feature[j]:
                            m[0][j] += 1
            win_count += 1

        # Window size in forward direction
        win_count = 0
        for i in range(index_value+1,len(A)):
            if win_count < window_size:
                if A[i] in sample_feature:
                    for j in range(len(sample_feature)):
                        if A[i] == sample_feature[j]:
                            m[0][j] += 1
            win_count += 1

    return m
```

In [3]:

```
# Implementation co variance (sample running and testing)
import tqdm

sample_feature = ['ABC','PQR','DEF']
A = ['ABC DEF IJK PQR','PQR KLM OPQ','LMN PQR XYZ ABC DEF PQR ABC']
window_size = 2
co_matrix_sample = np.zeros([3,3])

row = 0

# Iterating for each word i
for word_i in tqdm.tqdm_notebook(sample_feature):
    # Iterating to all corpus
#     print(word_i)
    for j in range(len(A)):
        index_ = []
        # Iterating each row/sentence to get the index value of 'present sample_feature'
        for k in range(len(A[j].split())):
            if word_i == A[j].split()[k]:
                index_.append(k)
        # If index is not empty that means there is some index present
        # For example for word i = 'ABC'
        # First I find the index of each row
        # If not present, then move to the new row.
        # If present, then store index in index_ variable
        # index_ for first row is [0]
        # Next iteration 'j', index_ for second row is []
        # Next iteration 'j', index_ for third row is [3,6]
        # Since, i got indexes, now i will take backward list and forward list with window_size except
its original index
```

```
        # value.
#         print(index_)
        if len(index_) != 0:
            # Based on index value, we find the row vector with size N (as no. of feature)
            for l in index_:
#                 print('perform: ',performrowupdation(sample_feature,A[j].split(),
#                                                      l,window_size,len(sample_feature))[0])
                co_matrix_sample[row] += performrowupdation(sample_feature,A[j].split(),
                                                            l,window_size,len(sample_feature))[0]
    co_matrix_sample[row][row] = 0
    row += 1
```

In [4]:

```
co_matrix_sample
```

Out[4]:

```
array([[0., 3., 3.],
       [3., 0., 2.],
       [3., 2., 0.]])
```

In [244]:

```
# Running real case scenario now
import tqdm

co_matrix_2k = np.zeros([2000,2000])
window_size = 5

row = 0

# Iterating for each word i
for word_i in tqdm.tqdm_notebook(essay_title_features):
    # Iterating to all corpus
    for j in range(len(e_t)):
        index_ = []
        # Iterating each row/sentence to get the index value of 'present sample_feature'
        for k in range(len(e_t[j].split())):
            if word_i == e_t[j].split()[k]:
                index_.append(k)
        # If index is not empty that means there is some index present
        # For example for word i = 'ABC'
        # First I find the index of each row
        # If not present, then move to the new row.
        # If present, then store index in index_ variable
        # index_ for first row is [0]
        # Next iteration 'j', index_ for second row is []
        # Next iteration 'j', index_ for third row is [3,6]
        # Since, i got indexes, now i will take backward list and forward list with window_size except
its original index
        # value.
        if len(index_) != 0:
            # Based on index value, we find the row vector with size N (as no. of feature)
            for l in index_:
                co_matrix_2k[row] += performrowupdation(essay_title_features,e_t[j].split(),
                                                        l,window_size,len(essay_title_features))[0]
    co_matrix_2k[row][row] = 0
    row += 1
```

In [245]:

```
pd.DataFrame(co_matrix_2k,index=essay_title_features,columns=essay_title_features).describe()
```

Out[245]:

	zumba	amplified	nannanencouraging	nannaneverything	ambiance	translator	nannanfind	nannanflexing	nannanfres

	zumba	amplified	nannanencouraging	nannaneverything	ambiance	translator	nannanfind	nannanflexing	nannanfree
count	2000.000000	2000.0	2000.0	2000.0	2000.0	2000.000000	2000.000000	2000.0	2000.
mean	0.000500	0.0	0.0	0.0	0.0	0.000500	0.000500	0.0	0.
std	0.022361	0.0	0.0	0.0	0.0	0.022361	0.022361	0.0	0.
min	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.
25%	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.
50%	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.
75%	0.000000	0.0	0.0	0.0	0.0	0.000000	0.000000	0.0	0.
max	1.000000	0.0	0.0	0.0	0.0	1.000000	1.000000	0.0	0.

8 rows × 2000 columns

In [246]:

```
pd.DataFrame(co_matrix_2k,index=essay_title_features,columns=essay_title_features).to_csv('co_matrix_2k
.csv')
```

2.3 Applying TruncatedSVD and Calculating Vectors for `essay` and `project_title`

In [0]:

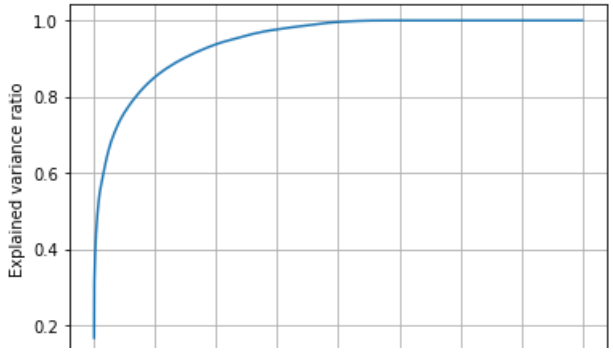
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

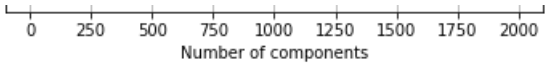
In [253]:

```
from sklearn.decomposition import TruncatedSVD

evr_ = []
for i in tqdm.tqdm_notebook(range(1,2000)):
    svd_ = TruncatedSVD(n_components=i, random_state=1)
    svd_.fit(co_matrix_2k)
    evr_.append(svd_.explained_variance_ratio_.sum())
```

In [254]:

```
plt.plot(np.arange(1,2000),evr_)
plt.grid()
plt.xlabel('Number of components')
plt.ylabel('Explained variance ratio')
plt.show()
```





Observation: Taking number of components = 750 which gives 95% data variance

In [255]:

```
svd_ = TruncatedSVD(n_components=750, random_state=1)
trun_cov = svd_.fit_transform(co_matrix_2k)
trun_cov.shape
```

Out[255]:

```
(2000, 750)
```

In [278]:

```
trun_cov = pd.DataFrame(trun_cov, index=essay_title_features, columns=essay_title_features[:750])
trun_cov.head(5)
```

Out[278]:

	zumba	amplified	nannanencouraging	nannaneverything	ambiance	translator	nannanfind	nannanflexing
zumba	2.751342e-15	4.239679e-16	-9.640680e-15	7.950231e-14	-4.433806e-14	-2.066622e-13	-1.288657e-12	-1.302673e-12
amplified	1.035817e-21	2.277168e-21	2.833534e-18	-4.858128e-18	-2.889368e-19	-2.194733e-19	-5.293533e-20	5.656580e-20
nannanencouraging	1.567503e-20	1.781012e-19	8.095063e-17	1.383671e-15	-2.604902e-16	-1.741117e-16	2.735859e-16	2.306788e-16
nannaneverything	5.171626e-21	1.495839e-19	-3.759551e-17	2.944236e-16	-2.015386e-15	8.908941e-16	-1.528352e-16	1.071009e-16
ambiance	6.019325e-22	3.255662e-21	8.511768e-17	1.485535e-18	9.018231e-16	4.082054e-18	-8.070656e-16	-2.172454e-15

5 rows × 750 columns

In [288]:

```
# Create dictonary
trun_cov_dict = {}
for i in range(2000):
    trun_cov_dict[str(essay_title_features[i])] = trun_cov.iloc[i,:].values
```

In [299]:

```
np.array([1,2,3,4])/4
```

Out[299]:

```
array([0.25, 0.5 , 0.75, 1.  ])
```

train essay and title

In [311]:

```
vec_essay = []
for i in tqdm.tqdm_notebook(range(tr.shape[0])):
    count = 0
    word_count = np.zeros([1,750])
```

```
    for j in tr['clean_essay'].values[i].split():
        if j in trun_cov_dict:
            word_count += trun_cov_dict[j]
        count += 1
    vec_essay.append(word_count/count)
```

In [312]:

```
vec_title = []
for i in tqdm.tqdm_notebook(range(tr.shape[0])):
    count = 0
    word_count = np.zeros([1,750])
    for j in tr['clean_project_title'].values[i].split():
        if j in trun_cov_dict:
            word_count += trun_cov_dict[j]
        count += 1
    vec_title.append(word_count/count)
```

In [313]:

```
vec_essay = np.array(vec_essay).squeeze()
vec_title = np.array(vec_title).squeeze()
```

In [314]:

```
vec_essay.shape,vec_title.shape
```

Out[314]:

```
((29998, 750), (29998, 750))
```

cv essay and title

In [350]:

```
cv_essay = []
for i in tqdm.tqdm_notebook(range(cv_.shape[0])):
    count = 0
    word_count = np.zeros([1,750])
    for j in cv_['clean_essay'].values[i].split():
        if j in trun_cov_dict:
            word_count += trun_cov_dict[j]
        count += 1
    cv_essay.append(word_count/count)
```

In [351]:

```
cv_title = []
for i in tqdm.tqdm_notebook(range(cv_.shape[0])):
    count = 0
    word_count = np.zeros([1,750])
    for j in cv_['clean_project_title'].values[i].split():
        if j in trun_cov_dict:
            word_count += trun_cov_dict[j]
        count += 1
    cv_title.append(word_count/count)
```

In [352]:

```
cv_essay = np.array(cv_essay).squeeze()
cv_title = np.array(cv_title).squeeze()
```

In [353]:

```
cv_essay.shape,cv_title.shape
```

Out[353]:

```
((10000, 750), (10000, 750))
```

test essay and title

In [388]:

```
ts_essay = []
for i in tqdm.tqdm_notebook(range(ts.shape[0])):
    count = 0
    word_count = np.zeros([1,750])
    for j in ts['clean_essay'].values[i].split():
        if j in trun_cov_dict:
            word_count += trun_cov_dict[j]
        count += 1
    ts_essay.append(word_count/count)
```

In [389]:

```
ts_title = []
for i in tqdm.tqdm_notebook(range(ts.shape[0])):
    count = 0
    word_count = np.zeros([1,750])
    for j in ts['clean_project_title'].values[i].split():
        if j in trun_cov_dict:
            word_count += trun_cov_dict[j]
        count += 1
    ts_title.append(word_count/count)
```

In [390]:

```
ts_essay = np.array(ts_essay).squeeze()
ts_title = np.array(ts_title).squeeze()
```

In [391]:

```
ts_essay.shape,ts_title.shape
```

Out[391]:

```
((10000, 750), (10000, 750))
```

2.4 Merge the features from step 3 and step 4

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [257]:

```
tr.head()
```

Out[257]:

Out[257]:

	teacher_prefix	school_state	teacher_number_of_previously_posted_projects	project_is_approved	price	quantity	clean_cat
102487	Ms.	NC	6	1	374.49	3	Math_S
34136	Mrs.	LA	5	1	440.22	2	Literacy_La
19849	Ms.	IL	5	1	499.99	1	AppliedL Mus
92095	Mrs.	CA	1	0	726.99	4	Math_S
20239	Ms.	IN	26	1	479.00	1	Literacy_La

In [261]:

```
tr_X = tr.drop(['project_is_approved'], axis=1)
tr_y = tr['project_is_approved'].values
tr_X.shape,tr_y.shape
```

Out[261]:

```
((29998, 10), (29998,))
```

In [361]:

```
cv_X = cv_.drop(['project_is_approved'], axis=1)
cv_y = cv_['project_is_approved'].values
cv_X.shape,cv_y.shape
```

Out[361]:

```
((10000, 10), (10000,))
```

In [387]:

```
ts_X = ts.drop(['project_is_approved'], axis=1)
ts_y = ts['project_is_approved'].values
ts_X.shape,ts_y.shape
```

Out[387]:

```
((10000, 10), (10000,))
```

Numerical Features Normailzation

In [262]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separately
```

```
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
# # For Numerical with train data

# ### 1) quantity
from sklearn.preprocessing import Normalizer
# # normalization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html

quantity_scalar = Normalizer()
quantity_scalar.fit(tr_X['quantity'].values.reshape(1,-1)) # finding the mean and standard deviation of this data
quantity_normalized = quantity_scalar.transform(tr_X['quantity'].values.reshape(1, -1))

# ### 2) price

# # the cost feature is already in numerical values,  we are going to represent the money, as numerical values within the range 0-1

price_scalar = Normalizer()
price_scalar.fit(tr_X['price'].values.reshape(1,-1)) # finding the mean and standard deviation of this data
price_normalized = price_scalar.transform(tr_X['price'].values.reshape(1, -1))

# ### 3) For teacher_number_of_previously_projects

# # We are going to represent the teacher_number_of_previously_posted_projects, as numerical values within the range 0-1

teacher_number_of_previously_posted_projects_scalar = Normalizer()
teacher_number_of_previously_posted_projects_scalar.fit(tr_X['teacher_number_of_previously_posted_projects'].values.reshape(1,-1)) # finding the mean and standard deviation of this data
teacher_number_of_previously_posted_projects_normalized = teacher_number_of_previously_posted_projects_scalar.transform(tr_X['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
```

In [263]:

```
print('Shape of quantity:', quantity_normalized.T.shape)
print('Shape of price:', price_normalized.T.shape)
print('Shape of teacher_number_of_previously_posted_projects:', teacher_number_of_previously_posted_projects_normalized.T.shape)
```

```
Shape of quantity: (29998, 1)
Shape of price: (29998, 1)
Shape of teacher_number_of_previously_posted_projects: (29998, 1)
```

In [264]:

```
quantity_normalized.T
```

Out[264]:

```
array([[0.0005719 ],
       [0.00038127],
       [0.00019063],
       ...,
       [0.04232075],
       [0.00076254],
       [0.00076254]])
```

In [265]:

```
price_normalized.T
```

Out[265]:

```
array([[0.00462813],
       [0.00544045],
       [0.00617912],
       ...,
       [0.00029685],
       [0.00296554],
       [0.00464654]])
```

In [266]:

```
teacher_number_of_previously_posted_projects_normalized.T
```

Out[266]:

```
array([[0.00116172],
       [0.0009681 ],
       [0.0009681 ],
       ...,
       [0.        ],
       [0.00135535],
       [0.00193621]])
```

In [323]:

```
cv_price = price_scalar.transform(cv_X['price'].values.reshape(1,-1))
cv_quantity = quantity_scalar.transform(cv_X['quantity'].values.reshape(1,-1))
cv_teacher_number_of_previously_posted_projects = \
teacher_number_of_previously_posted_projects_scalar.transform(cv_X['teacher_number_of_previously_posted
_projects'].\
                                                              values.reshape(1,-1))
```

In [392]:

```
ts_price = price_scalar.transform(ts_X['price'].values.reshape(1,-1))
ts_quantity = quantity_scalar.transform(ts_X['quantity'].values.reshape(1,-1))
ts_teacher_number_of_previously_posted_projects = \
teacher_number_of_previously_posted_projects_scalar.transform(ts_X['teacher_number_of_previously_posted
_projects'].\
                                                              values.reshape(1,-1))
```

Categorical Features

In [267]:

```
# For categorical with train data
# Please do the similar feature encoding with state, teacher_prefix and project_grade_category also
# One hot encoding for school state

### 1) school_state
print('==========================================================\n')
# Count Vectorize with vocuabulary contains unique code of school state and we are doing boolen BoW
vectorizer_school_state = CountVectorizer(vocabulary=tr_X['school_state'].unique(), lowercase=False, bi
nary=True)
vectorizer_school_state.fit(tr_X['school_state'].values)
print('List of feature in school_state',vectorizer_school_state.get_feature_names())

# Transform train data
school_state_one_hot = vectorizer_school_state.transform(tr_X['school_state'].values)
print("\nShape of school_state matrix after one hot encoding ",school_state_one_hot.shape)

### 2) project_subject_categories
print('==========================================================\n')
vectorizer_categories = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binar
y=True)
vectorizer_categories.fit(tr_X['clean_categories'].values)
print('List of features in project_subject_categories',vectorizer_categories.get_feature_names())

# Transform train data
categories_one_hot = vectorizer_categories.transform(tr_X['clean_categories'].values)
```

```
print("\nShape of project_subject_categories matrix after one hot encodig ",categories_one_hot.shape)

### 3) project_subject_subcategories
print('==========================================================\n')
vectorizer_subcategories = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False
, binary=True)
vectorizer_subcategories.fit(tr_X['clean_subcategories'].values)
print('List of features in project_subject_categories',vectorizer_subcategories.get_feature_names())

# Transform train data
subcategories_one_hot = vectorizer_subcategories.transform(tr_X['clean_subcategories'].values)
print("\nShape of project_subject_subcategories matrix after one hot encodig ",subcategories_one_hot.sh
ape)

### 4) project_grade_category
print('==========================================================\n')
# One hot encoding for project_grade_category

# Count Vectorize with vocuabulary contains unique code of project_grade_category and we are doing bool
en BoW
vectorizer_grade_category = CountVectorizer(vocabulary=tr_X['project_grade_category'].unique(), lowerca
se=False, binary=True)
vectorizer_grade_category.fit(tr_X['project_grade_category'].values)
print('List of features in project_grade_category',vectorizer_grade_category.get_feature_names())

# Transform train data
project_grade_category_one_hot = vectorizer_grade_category.transform(tr_X['project_grade_category'].val
ues)
print("\nShape of project_grade_category matrix after one hot encodig ",project_grade_category_one_hot.
shape)

### 5) teacher_prefix
print('==========================================================\n')
# One hot encoding for teacher_prefix

# Count Vectorize with vocuabulary contains unique code of teacher_prefix and we are doing boolen BoW
# Since some of the data is filled with nan. So we update the nan to 'None' as a string
# tr_X['teacher_prefix'] = tr_X['teacher_prefix'].fillna('None')
vectorizer_teacher_prefix = CountVectorizer(vocabulary=tr_X['teacher_prefix'].unique(), lowercase=False
, binary=True)
vectorizer_teacher_prefix.fit(tr_X['teacher_prefix'].values)
print('List of features in teacher_prefix',vectorizer_teacher_prefix.get_feature_names())

# Transform train data
teacher_prefix_one_hot = vectorizer_teacher_prefix.transform(tr_X['teacher_prefix'].values)
print("\nShape of teacher_prefix matrix after one hot encoding ",teacher_prefix_one_hot.shape)
```

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

List of feature in school_state ['NC', 'LA', 'IL', 'CA', 'IN', 'FL', 'SC', 'TX', 'MD', 'PA', 'WA', 'RI'
, 'AL', 'MN', 'WV', 'OH', 'KY', 'MI', 'UT', 'NY', 'NM', 'MS', 'WI', 'NJ', 'MA', 'CT', 'DC', 'GA', 'CO',
'NE', 'ME', 'AZ', 'DE', 'VA', 'NV', 'AK', 'OR', 'MO', 'AR', 'SD', 'TN', 'WY', 'OK', 'NH', 'ID', 'HI', '
MT', 'IA', 'KS', 'ND', 'VT']

Shape of school_state matrix after one hot encoding  (29998, 51)
==========================================================

List of features in project_subject_categories ['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts'
, 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']

Shape of project_subject_categories matrix after one hot encodig  (29998, 9)
==========================================================

List of features in project_subject_categories ['Economics', 'CommunityService', 'FinancialLiteracy', '
ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', '
Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other'
, 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeed
s', 'Literature_Writing', 'Mathematics', 'Literacy']

Shape of project_subject_subcategories matrix after one hot encodig  (29998, 30)
==========================================================

List of features in project_grade_category ['grades_3_5', 'grades_6_8', 'grades_prek_2', 'grades_9_12']

Shape of project_grade_category matrix after one hot encodig  (29998, 4)
```

Shape of project_grade_category matrix after one hot encoding (29998, 4)
==========

List of features in teacher_prefix ['Ms.', 'Mrs.', 'Mr.', 'Teacher', 'Dr.']

Shape of teacher_prefix matrix after one hot encoding (29998, 5)

In [324]:

```
# Transform categorical for cv data
cv_school_state = vectorizer_school_state.transform(cv_X['school_state'].values)
cv_project_subject_category = vectorizer_categories.transform(cv_X['clean_categories'].values)
cv_project_subject_subcategory = vectorizer_subcategories.transform(cv_X['clean_subcategories'].values)
cv_project_grade_category = vectorizer_grade_category.transform(cv_X['project_grade_category'].values)
cv_teacher_prefix = vectorizer_teacher_prefix.transform(cv_X['teacher_prefix'].values)
```

In [393]:

```
# Transform categorical for test data
ts_school_state = vectorizer_school_state.transform(ts_X['school_state'].values)
ts_project_subject_category = vectorizer_categories.transform(ts_X['clean_categories'].values)
ts_project_subject_subcategory = vectorizer_subcategories.transform(ts_X['clean_subcategories'].values)
ts_project_grade_category = vectorizer_grade_category.transform(ts_X['project_grade_category'].values)
ts_teacher_prefix = vectorizer_teacher_prefix.transform(ts_X['teacher_prefix'].values)
```

Count number of words in essay and title

In [269]:

```
tr_X.reset_index(drop=True,inplace=True)
tr_essay = []
# To calculate number of words, just take the length of each essay
for i in range(tr_X.shape[0]):
    tr_essay.append(len(tr_X['clean_essay'][i]))
tr_essay = np.array(tr_essay).reshape(-1,1)
tr_essay.shape
```

Out[269]:

(29998, 1)

In [270]:

```
tr_title = []
# To calculate number of words, just take the length of each title
for i in range(tr_X.shape[0]):
    tr_title.append(len(tr_X['clean_project_title'][i]))
tr_title = np.array(tr_title).reshape(-1,1)
tr_title.shape
```

Out[270]:

(29998, 1)

In [272]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()
tr_sen_essay = []

for i in tqdm_notebook(range(tr_X.shape[0])):
    ss = sid.polarity_scores(tr_X['clean_essay'][i])
    tr_sen_essay.append([ss['neg'],ss['neu'],ss['pos'],ss['compound']])
```

```
tr_sen_title = []

for i in tqdm.tqdm_notebook(range(tr_X.shape[0])):
    ss = sid.polarity_scores(tr_X['clean_project_title'][i])
    tr_sen_title.append([ss['neg'],ss['neu'],ss['pos'],ss['compound']])

tr_sen_essay = np.array(tr_sen_essay)
tr_sen_title = np.array(tr_sen_title)
# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

for cv essay and title

In [362]:

```
cv_X.reset_index(drop=True,inplace=True)
cv_essay_len = []
# To calculate number of words, just take the length of each essay
for i in range(cv_X.shape[0]):
    cv_essay_len.append(len(cv_X['clean_essay'][i]))
cv_essay_len = np.array(cv_essay_len).reshape(-1,1)
cv_essay_len.shape
```

Out[362]:

```
(10000, 1)
```

In [363]:

```
cv_title_len = []
# To calculate number of words, just take the length of each title
for i in range(cv_X.shape[0]):
    cv_title_len.append(len(cv_X['clean_project_title'][i]))
cv_title_len = np.array(cv_title_len).reshape(-1,1)
cv_title_len.shape
```

Out[363]:

```
(10000, 1)
```

In [328]:

```
sid = SentimentIntensityAnalyzer()
cv_sen_essay = []

for i in tqdm.tqdm_notebook(range(cv_X.shape[0])):
    ss = sid.polarity_scores(cv_X['clean_essay'][i])
    cv_sen_essay.append([ss['neg'],ss['neu'],ss['pos'],ss['compound']])

cv_sen_title = []

for i in tqdm.tqdm_notebook(range(cv_X.shape[0])):
    ss = sid.polarity_scores(cv_X['clean_project_title'][i])
    cv_sen_title.append([ss['neg'],ss['neu'],ss['pos'],ss['compound']])

cv_sen_essay = np.array(cv_sen_essay)
cv_sen_title = np.array(cv_sen_title)
# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

for test essay and title

In [394]:

```
ts_X.reset_index(drop=True,inplace=True)
ts_essay_len = []
# To calculate number of words, just take the length of each essay
```

```
# To calculate number of words, just take the length of each essay
for i in range(ts_X.shape[0]):
    ts_essay_len.append(len(ts_X['clean_essay'][i]))
ts_essay_len = np.array(ts_essay_len).reshape(-1,1)
ts_essay_len.shape
```

Out[394]:

```
(10000, 1)
```

In [395]:

```
ts_title_len = []
# To calculate number of words, just take the length of each title
for i in range(ts_X.shape[0]):
    ts_title_len.append(len(ts_X['clean_project_title'][i]))
ts_title_len = np.array(ts_title_len).reshape(-1,1)
ts_title_len.shape
```

Out[395]:

```
(10000, 1)
```

In [396]:

```
sid = SentimentIntensityAnalyzer()
ts_sen_essay = []

for i in tqdm.tqdm_notebook(range(ts_X.shape[0])):
    ss = sid.polarity_scores(ts_X['clean_essay'][i])
    ts_sen_essay.append([ss['neg'],ss['neu'],ss['pos'],ss['compound']])

ts_sen_title = []

for i in tqdm.tqdm_notebook(range(ts_X.shape[0])):
    ss = sid.polarity_scores(ts_X['clean_project_title'][i])
    ts_sen_title.append([ss['neg'],ss['neu'],ss['pos'],ss['compound']])

ts_sen_essay = np.array(ts_sen_essay)
ts_sen_title = np.array(ts_sen_title)
# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

In [ ]:

```
# Merge
```

In [415]:

```
# for train data
from scipy.sparse import hstack
tr_X = hstack((quantity_normalized.T, price_normalized.T, teacher_number_of_previously_posted_projects_
normalized.T, \
               school_state_one_hot, categories_one_hot, subcategories_one_hot, project_grade_category_o
ne_hot, \
               teacher_prefix_one_hot, tr_essay, tr_title, tr_sen_essay, tr_sen_title, vec_essay, vec_ti
tle))
tr_X.shape
```

Out[415]:

```
(29998, 1612)
```

In [416]:

```
# for cv data
from scipy.sparse import hstack
cv_X = hstack((cv_quantity.T, cv_price.T, cv_teacher_number_of_previously_posted_projects.T, \
```

```
            cv_school_state, cv_project_subject_category.toarray(), cv_project_subject_subcategory, c
v_project_grade_category, \
            cv_teacher_prefix, cv_essay_len, cv_title_len, cv_sen_essay, cv_sen_title, cv_essay, cv_t
itle))
cv_X.shape
```

Out[416]:

```
(10000, 1612)
```

In [420]:

```
np.save('cv_X.npy',cv_X.toarray())
np.save('tr_X.npy',tr_X.toarray())
```

In [417]:

```
# for test data
from scipy.sparse import hstack
ts_X = hstack((ts_quantity.T, ts_price.T, ts_teacher_number_of_previously_posted_projects.T, \
            ts_school_state, ts_project_subject_category.toarray(), ts_project_subject_subcategory, t
s_project_grade_category, \
            ts_teacher_prefix, ts_essay_len, ts_title_len, ts_sen_essay, ts_sen_title, ts_essay, ts_t
itle))
ts_X.shape
```

Out[417]:

```
(10000, 1612)
```

In [421]:

```
np.save('ts_X.npy',ts_X.toarray())
```

2.5 Apply XGBoost on the Final Features from the above section

https://xgboost.readthedocs.io/en/latest/python/python_intro.html

In [0]:

```
# No need to split the data into train and test(cv)
# use the Dmatrix and apply xgboost on the whole data
# please check the Quora case study notebook as reference

# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [422]:

```
# Load
tr_X = np.load('tr_X.npy',allow_pickle=True)
cv_X = np.load('cv_X.npy',allow_pickle=True)
ts_X = np.load('ts_X.npy',allow_pickle=True)
```

In [425]:

```
tr_X.shape,cv_X.shape,ts_X.shape
```

Out[425]:

```
((29998, 1612), (10000, 1612), (10000, 1612))
```

In [366]:

```
import sys
import math

import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score

# you might need to install this one
import xgboost as xgb
```

In [436]:

```
parameter = {'eta':[0.0001,0.001,0.01,0.1, 1, 10], \
'max_depth':[5,7,10]}
clf = xgb.XGBClassifier(random_state=1, class_weight='balanced')
```

In [437]:

```
tr_auc = []
cv_auc = []

for i in tqdm.tqdm_notebook(parameter['eta']):
    for j in parameter['max_depth']:
        clf = clf = xgb.XGBClassifier(random_state=1, class_weight='balanced',eta=i,max_depth=int(j))
        clf.fit(tr_X,tr_y)

        tr_pred = clf.predict(tr_X)
        cv_pred = clf.predict(cv_X)

        tr_auc.append(roc_auc_score(tr_y,tr_pred))
        cv_auc.append(roc_auc_score(cv_y,cv_pred))
```

In [441]:

```
for i in range(6):
    print('-'*10)
    print(parameter['eta'][i])
    print('-'*10)
    for j in range(3):
        print('Max Depth:',parameter['max_depth'][j])
        print('Train AUC: {} and CV AUC: {}'.format(tr_auc[i+j],cv_auc[i+j]))
```

```
----------
0.0001
----------
Max Depth: 5
Train AUC: 0.5310377138006244 and CV AUC: 0.5092513289271565
Max Depth: 7
Train AUC: 0.609772656345662 and CV AUC: 0.5169823500282965
Max Depth: 10
Train AUC: 0.7368275848862894 and CV AUC: 0.5300827674023769
----------
0.001
----------
Max Depth: 5
Train AUC: 0.609772656345662 and CV AUC: 0.5169823500282965
Max Depth: 7
Train AUC: 0.7368275848862894 and CV AUC: 0.5300827674023769
Max Depth: 10
Train AUC: 0.5310377138006244 and CV AUC: 0.5092513289271565
----------
0.01
----------
```

```
Max Depth: 5
Train AUC: 0.7368275848862894 and CV AUC: 0.5300827674023769
Max Depth: 7
Train AUC: 0.5310377138006244 and CV AUC: 0.5092513289271565
Max Depth: 10
Train AUC: 0.609772656345662 and CV AUC: 0.5169823500282965
----------
0.1
----------
Max Depth: 5
Train AUC: 0.5310377138006244 and CV AUC: 0.5092513289271565
Max Depth: 7
Train AUC: 0.609772656345662 and CV AUC: 0.5169823500282965
Max Depth: 10
Train AUC: 0.7368275848862894 and CV AUC: 0.5300827674023769
----------
1
----------
Max Depth: 5
Train AUC: 0.609772656345662 and CV AUC: 0.5169823500282965
Max Depth: 7
Train AUC: 0.7368275848862894 and CV AUC: 0.5300827674023769
Max Depth: 10
Train AUC: 0.5310377138006244 and CV AUC: 0.5092513289271565
----------
10
----------
Max Depth: 5
Train AUC: 0.7368275848862894 and CV AUC: 0.5300827674023769
Max Depth: 7
Train AUC: 0.5310377138006244 and CV AUC: 0.5092513289271565
Max Depth: 10
Train AUC: 0.609772656345662 and CV AUC: 0.5169823500282965
```

Found eta = 0.1 to be reasonable
Let fix eta = 0.1 and cary with n_estimators

In [442]:

```
bckup_tr = tr_auc
bckup_cv = cv_auc
```

In [444]:

```
parameter = {'n_estimators':[2,3,5,10,20], \
'max_depth':[5,7,10]}
clf = xgb.XGBClassifier(random_state=1, class_weight='balanced')

tr_auc = []
cv_auc = []

for i in tqdm.tqdm_notebook(parameter['n_estimators']):
    for j in parameter['max_depth']:
        clf = clf = xgb.XGBClassifier(random_state=1, class_weight='balanced',eta=0.1 \
                                      , n_estimators=i,max_depth=int(j))
        clf.fit(tr_X,tr_y)

        tr_pred = clf.predict(tr_X)
        cv_pred = clf.predict(cv_X)

        tr_auc.append(roc_auc_score(tr_y,tr_pred))
        cv_auc.append(roc_auc_score(cv_y,cv_pred))
```

In [446]:

```
for i in range(5):
    print('-'*10)
    print(parameter['n_estimators'][i])
    print('-'*10)
    for j in range(3):
        print('Max Depth:',parameter['max_depth'][j])
        print('Train AUC: {} and CV AUC: {}'.format(tr_auc[i+j],cv_auc[i+j]))
```

```
----------
2
----------
Max Depth: 5
Train AUC: 0.5029014218992197 and CV AUC: 0.4997062969924812
Max Depth: 7
Train AUC: 0.5125758392102814 and CV AUC: 0.5017413746058695
Max Depth: 10
Train AUC: 0.5669102469536449 and CV AUC: 0.5207404852857952
----------
3
----------
Max Depth: 5
Train AUC: 0.5125758392102814 and CV AUC: 0.5017413746058695
Max Depth: 7
Train AUC: 0.5669102469536449 and CV AUC: 0.5207404852857952
Max Depth: 10
Train AUC: 0.5027616132873857 and CV AUC: 0.49976503759398494
----------
5
----------
Max Depth: 5
Train AUC: 0.5669102469536449 and CV AUC: 0.5207404852857952
Max Depth: 7
Train AUC: 0.5027616132873857 and CV AUC: 0.49976503759398494
Max Depth: 10
Train AUC: 0.5108258422592212 and CV AUC: 0.4992458464710163
----------
10
----------
Max Depth: 5
Train AUC: 0.5027616132873857 and CV AUC: 0.49976503759398494
Max Depth: 7
Train AUC: 0.5108258422592212 and CV AUC: 0.4992458464710163
Max Depth: 10
Train AUC: 0.5601024304567324 and CV AUC: 0.5203147738297357
----------
20
----------
Max Depth: 5
Train AUC: 0.5108258422592212 and CV AUC: 0.4992458464710163
Max Depth: 7
Train AUC: 0.5601024304567324 and CV AUC: 0.5203147738297357
Max Depth: 10
Train AUC: 0.5017085322281368 and CV AUC: 0.49988251879699247
```

Found n_estimators= 10 to be reasonable

In [448]:

```
clf = clf = xgb.XGBClassifier(random_state=1, class_weight='balanced',eta=0.1 \
                                    , n_estimators=10,max_depth=10)
clf.fit(tr_X,tr_y)
```

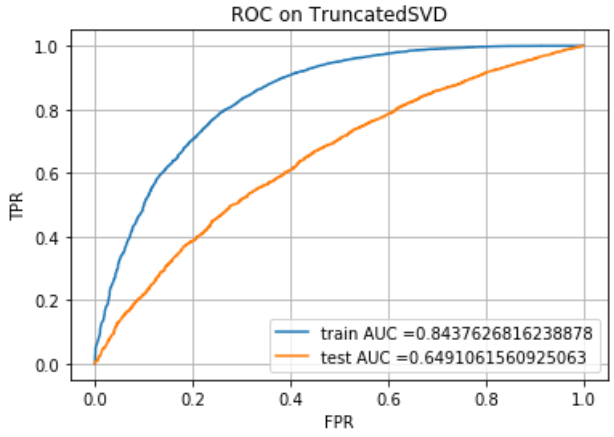
Out[448]:

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              eta=0.1, gamma=0, learning_rate=0.1, max_delta_step=0,
              max_depth=10, min_child_weight=1, missing=None, n_estimators=10,
              n_jobs=1, nthread=None, objective='binary:logistic',
              random_state=1, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
              seed=None, silent=None, subsample=1, verbosity=1)
```

In [451]:

```
y_train_pred = clf.predict_proba(tr_X)[:,1]
y_test_pred = clf.predict_proba(ts_X)[:,1]
train_fpr, train_tpr, tr_thresholds = roc_curve(tr_y, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(ts_y, 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("FPR")
plt.ylabel("TPR")
plt.title("ROC on TruncatedSVD")
plt.grid()
plt.show()
```



In [460]:

```
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (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))
    return t

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

In [461]:

```
len(predict_with_best_t(y_train_pred, best_t))
```

Out[461]:

29998

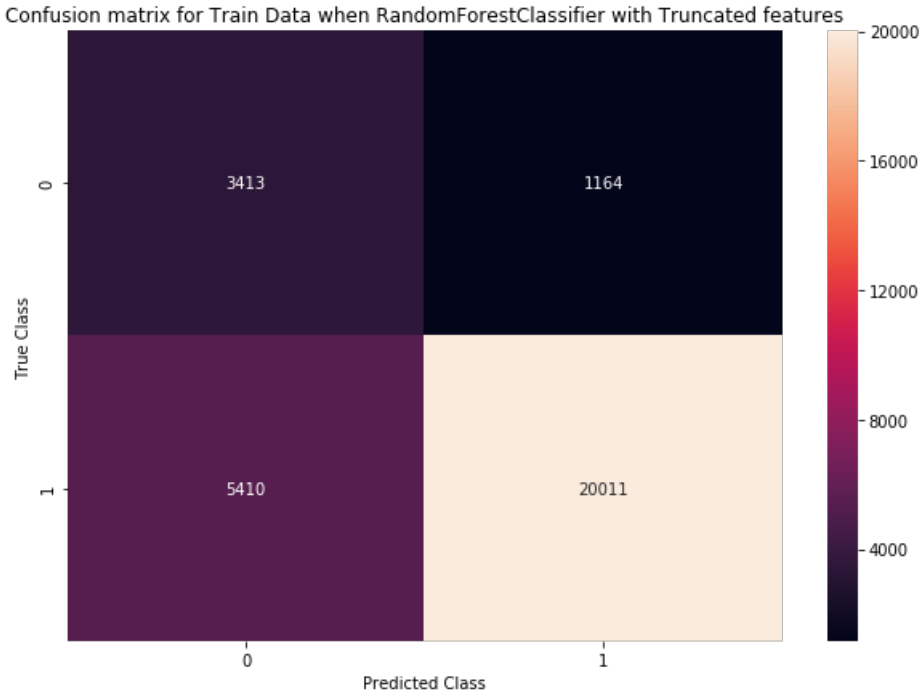
In [462]:

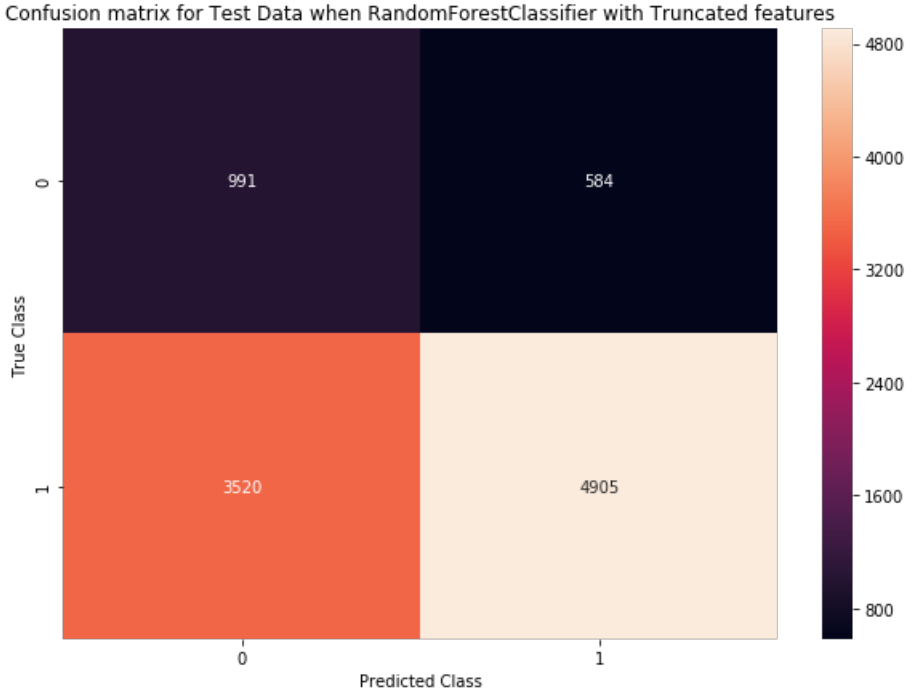
```
feature_names = 'Truncated'
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
cm = metrics.confusion_matrix(tr_y, predict_with_best_t(y_train_pred, best_t))
plt.figure(figsize = (10,7))
sns.heatmap(cm, annot=True, fmt="d")
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title('Confusion matrix for Train Data when RandomForestClassifier with {0} features'.format(featur
e_names))

print("Test confusion matrix")
cm = metrics.confusion_matrix(ts_y, predict_with_best_t(y_test_pred, best_t))
plt.figure(figsize = (10,7))
sns.heatmap(cm, annot=True, fmt="d")
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title('Confusion matrix for Test Data when RandomForestClassifier with {0} features'.format(feature
_names))
```

```
the maximum value of tpr*(1-fpr) 0.5869911279588114 for threshold 0.709
Train confusion matrix
Test confusion matrix
```

Out[462]:

```
Text(0.5, 1.0, 'Confusion matrix for Test Data when RandomForestClassifier with Truncated features')
```





3. Conclusion

1. From the confusion Matrix in Train and Test Data, we found that more fractional data whose label = 1 (that is, project is approved) has mispredict more than the data whose label = 0 (project is not approved)
2. Model performed as follows table below:

In [469]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ['Features','Model','eta','n_estimators','max_depth','train auc','test auc']
x.add_row(['Co-occurence Matrix TruncatedSVD','XGBoost','0.1','10','10','0.84','0.65'])
print(x)
```

```
+---------------------------------+---------+-----+--------------+-----------+-----------+----------+
|             Features            |  Model  | eta | n_estimators | max_depth | train auc | test auc |
+---------------------------------+---------+-----+--------------+-----------+-----------+----------+
| Co-occurence Matrix TruncatedSVD | XGBoost | 0.1 |      10      |     10    |    0.84   |   0.65   |
+---------------------------------+---------+-----+--------------+-----------+-----------+----------+
```

In [ ]: