DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

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

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

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

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

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

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

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

from tqdm import tqdm
import os

from chart_studio.plotly import plot, iplot
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.1 Reading Data

In [2]:

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

In [3]:

```
print("Number of data points in train data", project_data.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
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [5]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in
-one-step
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')
```

In [6]:

```
# #Sampling down the data
project_data = project_data.sample(frac=0.5)
```

1.2 preprocessing of `project_subject_categories`

In [7]:

```
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 & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [8]:

```
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 & H
unger"]
```

```
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

preprocessing of `teacher_prefix`

In [9]:

```
#NaN values in techer prefix will create a problem while encoding,so we replace NaN values with th
e mode of that particular column
#removing dot(.) since it is a special character
mode_of_teacher_prefix = project_data['teacher_prefix'].value_counts().index[0]


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

In [10]:

```
prefixes = []

for i in project_data['teacher_prefix']:
    prefixes.append(i.replace(".", ""))
```

In [11]:

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

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

Out[11]:

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

preprocessing of `project_grade_category`

In [12]:

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

project_grade_category = []

for i in project_data["project_grade_category"]:

    project_grade_category.append(i.replace(" ", "_").replace("-", "_"))
```

In [13]:

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

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

Out[13]:

```
array(['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2'],
      dtype=object)
```

1.3 Text preprocessing

In [14]:

```
# 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 [15]:

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

Out[15]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	project_essay_
46768	40859	p102887	ee12630540f08892aa2858de76283220	MN	2016-09-04 17:55:43	Making Connections Through Science in 5th Grade!	5th grade is time whe students ar making c.
45087	53381	p116886	b1bb6db4d1f80c65438f4955771900dd	NY	2017-01-08 20:21:59	Studying Biotechnology	Pleasan unassumin teenagers I have who tot.

In [16]:

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

In [17]:

```
# 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)
```

```
5th grade is a time when students are making connections of what they are learning in the classroo
m to what is happening in the world. To help make this connection more enriching and meaningful, w
e are wanting to expose our students to engaging texts found in Scholastic's \"SuperScience\" maga
zine. \r\nOur students yearn for knowledge. They love asking questions and making connections to w
hat they know and to what they are learning. Our school has many students who are free and reduced
lunch, yet this does not stop our expectations for exposing our students to a high, rigorous educa
tion.\r\n\r\nThis year in 5th grade, we will be using \"SuperScience\"  as a part of our
curriculum to help bridge what we are learning and show examples of what we are learning to what i
s going on in the world around us. Our hope is that students will extend their knowledge of real-w
```

orld science and social studies, as well as become more comfortable with reading nonfiction texts. \r\nOur goal for our students is for them to become aware of science in the world around us. We wa nt them to not only know what is happening in their life, but in the lives of others in our city, state, country, and world. Because many of our students come from other countries, we are also hop ing to build that bridge of connecting where they come from to where they are now to where they might go in the future. \r\nnannan
==========
Our school is located in a small rural community of 400. The school serves PreK-12 grade students in the surrounding farming and ranching areas. 53% of our students receive free or reduced lunch. \r\n\r\nWhile our school is small, the spirit, motivation, and drive of our students is far reachi ng. Our FFA program has gone to Nationals the past several years, our High School sports teams co mpete in State-Level tournaments, and our students have created a thriving garden that feeds our s chool and community. \r\n\r\nHowever, as our school is small and rural, funding for resources is scarce. Our publications program (yearbook, newspaper, digital media) is stuck in the dark ages w ith outdated, old, barley working cameras.Students currently learn photographic skills on outdated cameras that don't work reliably. These two cameras will allow students to explore life and culture using photography. \r\n\r\nStudents will not only expand their photographic skills but al so gain knowledge and real world experience in publications, digital media, composition, use of li ght, and graphic design. Our classes integrate core subjects- looking at the science of light, his tory of photojournalism and photography, and writing and developing digital stories. \r\n\r\nThe d igital cameras will assist students in gaining valuable life and work skills that they can use bey ond the classroom. They will also provide students with the opportunity to document school life.n annan
==========
My third graders are energetic learners. They come to school ready to learn with inquisitive minds that are like sponges. My students are eager to learn.\r\nI teach at a large urban elementary Tile 1 school in which one hundred percent of the students receive free breakfast and l unch. Many of my students come from two-parent homes in which both parents work trying to make end meet. Some of my students come from single-parent homes. I just want to provide the best educational experiences possible for all m students.The materials that I am requesting will help m y students to be better organized. The folders and sheet protectors will help students to keep im portant papers safe. The pencil pouches will help the students keep their pencils, pens, and mark ers and in one place and not roll off their desks. Students become frustrated when they cannot fin d certain papers or when their pens or pencils roll right off their desks in the middle of a lesson.\r\nYour donations will help my students learn to become better organized. Organization is a skill necessary for future education and any job they may hold in years to come.nannan
==========
I have 19 students in my classroom and each one of them is very special. They have grown so much s ince the beginning of the year. Now they are ready to go to Kindergarten. However, we still have a few more weeks in school and I will make those weeks count. I have a fantastic class! They are e ager to learn and always willing to work. \r\nMy students come from low income families and it is always a struggle to get more supplies. As we are approaching the end of the year we are running o ut of centers and fun activities to do. I believe my scholars had work so hard the whole year that I want to keep that pace until the end of the year. \r\nThey deserve a good quality instruction ev eryday of the year to target their weakness and improve their strengths. \r\nI picked these centers to reinforced their skills in mathematics and reading. I have 19 bilingual students in my classroom and all of them are unique and have different learning styles. \r\nMy mission is to targ et all their needs before the end of the year. I would like to have the Spanish File Folder Game to work with my high group because they are ready for a bigger challenge. The Alphabet Feel & Find Sensory Tub will make all my students so excited to go to the ABC center. The Write and Wipe Alp habet is a fun way to keep learning and reinforcing their letter knowledge. We need this centers t o continue with our learning and to keep them engage until the last day of school. \r\nIn math, w e have used all the centers that I already have. I know for sure my students want to use different activities to complete their math centers. The Airplane Counting Box and the Scoop and Count Ice C ream would be a fantastic center that my children will enjoy. These centers will motivate my scho lars to learn number recognition and to count. \r\nI am writing this project hopping to be able to have these resources in my classroom. I really want to keep all my students engage until the end o f the school year. My students had work so hard and I know they are Kinder-ready, but they need to keep practicing all their skills to get their feet wet for next year.nannan
==========

In [18]:

```
# 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 [19]:

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

```
I have 19 students in my classroom and each one of them is very special. They have grown so much s
ince the beginning of the year.  Now they are ready to go to Kindergarten. However, we still have
a few more weeks in school and I will make those weeks count. I have a fantastic class! They are e
ager to learn and always willing to work. \r\nMy students come from low income families and it is
always a struggle to get more supplies. As we are approaching the end of the year we are running o
ut of centers and fun activities to do. I believe my scholars had work so hard the whole year that
I want to keep that pace until the end of the year. \r\nThey deserve a good quality instruction ev
eryday of the year to target their weakness and improve their strengths. \r\nI picked these
centers to reinforced their skills in mathematics and reading. I have 19 bilingual students in my
classroom and all of them are unique and have different learning styles. \r\nMy mission is to targ
et all their needs before the end of the year.  I would like to have the Spanish File Folder Game
to work with my high group because they are ready for a bigger challenge. The Alphabet Feel & Find
Sensory Tub will make all my students so excited to go to the ABC center.   The Write and Wipe Alp
habet is a fun way to keep learning and reinforcing their letter knowledge. We need this centers t
o continue with our learning  and to keep them engage until the last day of school. \r\nIn math, w
e have used all the centers that I already have. I know for sure my students want to use different
activities to complete their math centers. The Airplane Counting Box and the Scoop and Count Ice C
ream would be a fantastic center that my children will enjoy.  These centers will motivate my scho
lars to learn number recognition and to count. \r\nI am writing this project hopping to be able to
have these resources in my classroom. I really want to keep all my students engage until the end o
f the school year. My students had work so hard and I know they are Kinder-ready, but they need to
keep practicing all their skills to get their feet wet for next year.nannan
==================================================
```

In [20]:

```
# \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)
```

```
I have 19 students in my classroom and each one of them is very special. They have grown so much s
ince the beginning of the year.  Now they are ready to go to Kindergarten. However, we still have
a few more weeks in school and I will make those weeks count. I have a fantastic class! They are e
ager to learn and always willing to work.   My students come from low income families and it is al
ways a struggle to get more supplies. As we are approaching the end of the year we are running out
of centers and fun activities to do. I believe my scholars had work so hard the whole year that I
want to keep that pace until the end of the year.   They deserve a good quality instruction
everyday of the year to target their weakness and improve their strengths.   I picked these
centers to reinforced their skills in mathematics and reading. I have 19 bilingual students in my
classroom and all of them are unique and have different learning styles.   My mission is to target
all their needs before the end of the year.  I would like to have the Spanish File Folder Game to
work with my high group because they are ready for a bigger challenge. The Alphabet Feel & Find Se
nsory Tub will make all my students so excited to go to the ABC center.   The Write and Wipe Alpha
bet is a fun way to keep learning and reinforcing their letter knowledge. We need this centers to
continue with our learning  and to keep them engage until the last day of school.   In math, we ha
ve used all the centers that I already have. I know for sure my students want to use different act
ivities to complete their math centers. The Airplane Counting Box and the Scoop and Count Ice Crea
m would be a fantastic center that my children will enjoy.  These centers will motivate my
scholars to learn number recognition and to count.   I am writing this project hopping to be able
to have these resources in my classroom. I really want to keep all my students engage until the en
d of the school year. My students had work so hard and I know they are Kinder-ready, but they need
to keep practicing all their skills to get their feet wet for next year.nannan
```

In [21]:

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

```
I have 19 students in my classroom and each one of them is very special They have grown so much si
nce the beginning of the year Now they are ready to go to Kindergarten However we still have a few
more weeks in school and I will make those weeks count I have a fantastic class They are eager to
learn and always willing to work My students come from low income families and it is always a stru
ggle to get more supplies As we are approaching the end of the year we are running out of centers
and fun activities to do I believe my scholars had work so hard the whole year that I want to keep
that pace until the end of the year They deserve a good quality instruction everyday of the year t
o target their weakness and improve their strengths I picked these centers to reinforced their
skills in mathematics and reading I have 19 bilingual students in my classroom and all of them are
unique and have different learning styles My mission is to target all their needs before the end o
f the year I would like to have the Spanish File Folder Game to work with my high group because th
ey are ready for a bigger challenge The Alphabet Feel Find Sensory Tub will make all my students s
o excited to go to the ABC center The Write and Wipe Alphabet is a fun way to keep learning and re
inforcing their letter knowledge We need this centers to continue with our learning and to keep th
em engage until the last day of school In math we have used all the centers that I already have I
know for sure my students want to use different activities to complete their math centers The Airp
lane Counting Box and the Scoop and Count Ice Cream would be a fantastic center that my children w
ill enjoy These centers will motivate my scholars to learn number recognition and to count I am wr
iting this project hopping to be able to have these resources in my classroom I really want to
keep all my students engage until the end of the school year My students had work so hard and I kn
ow they are Kinder ready but they need to keep practicing all their skills to get their feet wet f
or next year nannan
```

In [22]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', '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', "d
esn'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 [23]:

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

In [24]:

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

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

```
100%|██████████| 54624/54624 [00:22<00:00, 2463.53it/s]
```

In [25]:

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

Out[25]:

```
'19 students classroom one special grown much since beginning year ready go kindergarten however s
till weeks school make weeks count fantastic class eager learn always willing work students come l
ow income families always struggle get supplies approaching end year running centers fun activitie
s believe scholars work hard whole year want keep pace end year deserve good quality instruction e
veryday year target weakness improve strengths picked centers reinforced skills mathematics readin
g 19 bilingual students classroom unique different learning styles mission target needs end year w
ould like spanish file folder game work high group ready bigger challenge alphabet feel find senso
ry tub make students excited go abc center write wipe alphabet fun way keep learning reinforcing l
etter knowledge need centers continue learning keep engage last day school math used centers alrea
dy know sure students want use different activities complete math centers airplane counting box sc
oop count ice cream would fantastic center children enjoy centers motivate scholars learn number r
ecognition count writing project hopping able resources classroom really want keep students engage
end school year students work hard know kinder ready need keep practicing skills get feet wet next
year'
```

In [26]:

```
# after preprocesing

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

1.4 Preprocessing of `project_title`

In [27]:

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

In [28]:

```
# similarly you can preprocess the titles also

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

```
100%|██████████| 54624/54624 [00:00<00:00, 60297.24it/s]
```

In [29]:

```
preprocessed_titles[30000]
```

Out[29]:

```
'thirsty minds need drink osmo'
```

In [30]:

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

1.5 Preparing data for models

In [31]:

```
project_data.columns
```

Out[31]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'teacher_prefix', 'project_grade_category', 'essay', 'clean_essays',
       'clean_titles'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.2.3 Using Pretrained Models: Avg W2V

In [38]:

```
'''
# 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-sa
ve-and-load-variables-in-python/

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


'''
```

Out[38]:

```
'\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        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love 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 tha
t 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]\n
print("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 pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [39]:

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

1.5.3 Vectorizing Numerical features

Computing Sentiment Scores

In [50]:

```
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 w
ith the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelli
gences i use a wide range\
of techniques to help all my students succeed students in my class come from a variety of differen
t backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a carin
g community of successful \
learners which can be seen through collaborative student project based learning in and out of the
classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities to practice
a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the ki
ndergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role pla
y in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food
i will take their idea \
and create common core cooking lessons where we learn important math and writing concepts while co
oking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that went into maki
ng the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project woul
d expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade apple
sauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create our own cook
books to be printed and \
shared with families students will gain math and literature skills as well as a life long enjoymen
t 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
```

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

In [51]:

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

# import nltk
nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()
sentiment_titles=[]

for sentance in tqdm(project_data['essay'].values):
    ss = sid.polarity_scores(sentance)
    sentiment_titles.append(ss)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/ubuntu/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
100%|██████████| 54624/54624 [01:35<00:00, 573.47it/s]
```

In [52]:

```
sentiment_neg=[]
sentiment_neu=[]
```

```
sentiment_pos=[]
sentiment_compound=[]

for i in sentiment_titles:
    for j,k in i.items():
        if(j=='neg'):
            sentiment_neg.append(k)
        else:
            if(j=='neu'):
                sentiment_neu.append(k)
            else:
                if(j=='pos'):
                    sentiment_pos.append(k)
                else:
                    if(j=='compound'):
                        sentiment_compound.append(k)
```

In [53]:

```
#adding each sentiment score in a new column in the df
project_data['sentiment_neg'] = sentiment_neg
project_data['sentiment_neu'] = sentiment_neu
project_data['sentiment_pos'] = sentiment_pos
project_data['sentiment_compound'] = sentiment_compound
```

In [54]:

```
#https://stackoverflow.com/questions/49984905/count-number-of-words-per-row
project_data['words_title'] = project_data['project_title'].str.split().str.len()
```

In [55]:

```
#https://stackoverflow.com/questions/49984905/count-number-of-words-per-row
project_data['words_essay'] = project_data['essay'].str.split().str.len()
```

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 [56]:

```
import sys
import math

import numpy as np
#from sklearn.grid_search import GridSearchCV
from sklearn.model_selection 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, ve
rbose_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)
```

Out[56]:

```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=<__main__.XGBoostClassifier object at 0x7f7fac577390>,
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'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]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

In [57]:

```
clf.best_params_
```

Out[57]:

```
{'colsample_bytree': 0.9,
 'eta': 0.05,
 'max_depth': 6,
 'num_boost_round': 100,
 'subsample': 0.9}
```

2. TruncatedSVD

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

In [58]:

```
# 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
project_data["titles_essays"] = project_data["clean_titles"].map(str) +\
                        project_data["clean_essays"].map(str)
```

In [59]:

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

Out[59]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	project_resou
46768	40859	p102887	ee12630540f08892aa2858de76283220	MN	2016-09-04 17:55:43	making connections through science in 5th grade!	My interactive
45087	53381	p116886	b1bb6db4d1f80c65438f4955771900dd	NY	2017-01-08 20:21:59	studying biotechnology	My stuc apparatus

2 rows × 25 columns

In [60]:

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

In [61]:

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

In [62]:

```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(24520, 24) (24520,)
(12078, 24) (12078,)
(18026, 24) (18026,)
====================================================================================================
```

In [63]:

```
print(X_train.columns)
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'clean_categories', 'clean_subcategories', 'teacher_prefix',
       'project_grade_category', 'essay', 'clean_essays', 'clean_titles',
       'sentiment_neg', 'sentiment_neu', 'sentiment_pos', 'sentiment_compound',
       'words_title', 'words_essay', 'titles_essays'],
      dtype='object')
```

In [64]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vect = TfidfVectorizer(ngram_range = (1,1) , max_features = 2000)
tfidf_train = tfidf_vect.fit_transform (X_train['titles_essays'])
```

In [65]:

```
top_2000 = tfidf_vect.get_feature_names()
```

2.2 Computing Co-occurance matrix

In [66]:

```
# https://github.com/Manish-12/Truncated-SVD-on-Amazon-fine-food-reviews-
/blob/master/Truncated%20SVD.ipynb
# 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
from tqdm import tqdm
n_neighbor = 5
occ_matrix_2000 = np.zeros((2000,2000))
for row in tqdm(X_train["titles_essays"].values):
    words_in_row = row.split()
    for index,word in enumerate(words_in_row):
        if word in top_2000:
            for j in range(max(index-n_neighbor,0),min(index+n_neighbor,len(words_in_row)-1) + 1):
                if words_in_row[j] in top_2000:
```

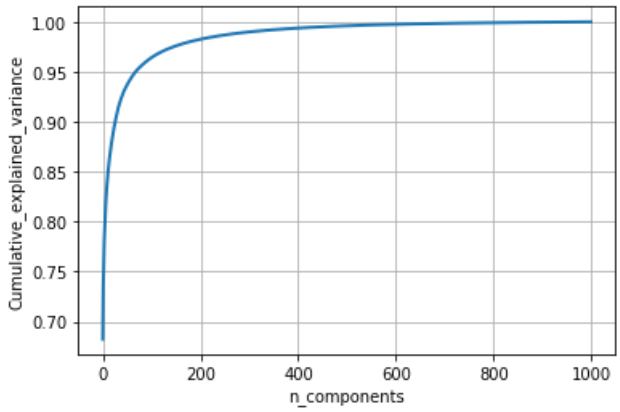
```
                    occ_matrix_2000[top_2000.index(word),top_2000.index(words_in_row[j])] += 1
                else:
                    pass
            else:
                pass
```

```
100%|██████████| 24520/24520 [19:29<00:00, 21.98it/s]
```

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

In [67]:

```
#https://chrisalbon.com/machine_learning/feature_engineering/select_best_number_of_components_in_t

from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import StandardScaler
svd = TruncatedSVD(n_components = 1000)
svd_2000 = svd.fit_transform(occ_matrix_2000)

percentage_var_explained = svd.explained_variance_ / np.sum(svd.explained_variance_);
cum_var_explained = np.cumsum(percentage_var_explained)
plt.figure(figsize=(6, 4))

plt.clf()
plt.plot(cum_var_explained, linewidth=2)
plt.axis('tight')
plt.grid()
plt.xlabel('n_components')
plt.ylabel('Cumulative_explained_variance')
plt.show()
```



In [68]:

```
svd = TruncatedSVD(n_components = 150)
svd_2000 = svd.fit_transform(occ_matrix_2000)
```

In [69]:

```
my_dict={}
# print(type(my_dict))
my_dict = dict.fromkeys(top_2000 , 1)
```

In [70]:

```
svd_2000_list=svd_2000.tolist()
```

In [71]:

```
my_dict = { i : svd_2000_list[i] for i in range(0, len(svd_2000_list) ) }
```

In [72]:

```
first2pairs = {k: my_dict[k] for k in list(my_dict)[:1]}
```

In [73]:

```
print(len(my_dict))
print(svd_2000.shape)
```

```
2000
(2000, 150)
```

In [77]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train["titles_essays"].values)
# 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 [78]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train["titles_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 tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_essays.append(vector)

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

```
100%|██████████| 24520/24520 [00:33<00:00, 741.41it/s]

24520
300
```

In [79]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train["titles_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 tf
idf 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
    cv_tfidf_w2v_essays.append(vector)
```

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

```
100%|██████████| 24520/24520 [00:32<00:00, 750.32it/s]
```

```
24520
300
```

In [80]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train["titles_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 tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_essays.append(vector)

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

```
100%|██████████| 24520/24520 [00:31<00:00, 769.43it/s]
```

```
24520
300
```

In [100]:

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

train_categories_one_hot = vectorizer_cat.transform(X_train['clean_categories'].values)
cv_categories_one_hot = vectorizer_cat.transform(X_cv['clean_categories'].values)
test_categories_one_hot = vectorizer_cat.transform(X_test['clean_categories'].values)


print(vectorizer_cat.get_feature_names())
print("Shape of training data matrix after one hot encoding ",train_categories_one_hot.shape)
print("Shape of CV data matrix after one hot encoding ",cv_categories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",test_categories_one_hot.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of training data matrix after one hot encoding  (24520, 9)
Shape of test data matrix after one hot encoding  (18026, 9)
```

In [101]:

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

```
train_subcategories_one_hot = vectorizer_subcat.transform(X_train['clean_subcategories'].values)
cv_subcategories_one_hot = vectorizer_subcat.transform(X_cv['clean_subcategories'].values)
test_subcategories_one_hot = vectorizer_subcat.transform(X_test['clean_subcategories'].values)


print(vectorizer_subcat.get_feature_names())


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

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
Shape of train data matrix after one hot encoding  (24520, 30)
Shape of CV data matrix after one hot encoding  (12078, 30)
Shape of test data matrix after one hot encoding  (18026, 30)
```

In [81]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(X_train["titles_essays"])      #Fit has to be on Train data

train_essay_tfidf = vectorizer_tfidf_essay.transform(X_train["titles_essays"].values)
cv_essay_tfidf = vectorizer_tfidf_essay.transform(X_cv["titles_essays"].values)
test_essay_tfidf = vectorizer_tfidf_essay.transform(X_test["titles_essays"].values)

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

```
Shape of train data matrix after one hot encoding  (24520, 9440)
Shape of test data matrix after one hot encoding  (12078, 9440)
Shape of test data matrix after one hot encoding  (18026, 9440)
```

In [ ]:

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

In [ ]:

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

In [82]:

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

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



print(vectorizer_grade.get_feature_names())

train_project_grade_category_one_hot = vectorizer_grade.transform(X_train['project_grade_category'
].values)
cv_project_grade_category_one_hot =
vectorizer_grade.transform(X_cv['project_grade_category'].values)
test_project_grade_category_one_hot = vectorizer_grade.transform(X_test['project_grade_category'].
values)


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

```
ape)
print("Shape of cv data matrix after one hot encoding ",cv_project_grade_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",test_project_grade_category_one_hot.shap
e)
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of train data matrix after one hot encoding  (24520, 4)
Shape of cv data matrix after one hot encoding  (12078, 4)
Shape of test data matrix after one hot encoding  (18026, 4)
```

In [83]:

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

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



print(vectorizer_school_state.get_feature_names())

train_school_state_category_one_hot = vectorizer_school_state.transform(X_train['school_state'].va
lues)

cv_school_state_category_one_hot = vectorizer_school_state.transform(X_cv['school_state'].values)

test_school_state_category_one_hot =
vectorizer_school_state.transform(X_test['school_state'].values)


print("Shape of train data matrix after one hot encoding ",train_school_state_category_one_hot.sha
pe)
print("Shape of cv data matrix after one hot encoding ",cv_school_state_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",test_school_state_category_one_hot.shape
)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv
', 'wy']
Shape of train data matrix after one hot encoding  (24520, 51)
Shape of train data matrix after one hot encoding  (12078, 51)
Shape of test data matrix after one hot encoding  (18026, 51)
```

In [84]:

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


print(vectorizer_prefix.get_feature_names())


train_teacher_prefix_categories_one_hot = vectorizer_prefix.transform(X_train['teacher_prefix'].va
lues.astype("U"))
cv_teacher_prefix_categories_one_hot = vectorizer_prefix.transform(X_cv['teacher_prefix'].values.a
stype("U"))
test_teacher_prefix_categories_one_hot =
vectorizer_prefix.transform(X_test['teacher_prefix'].values.astype("U"))

print("Shape of train data matrix after one hot encoding ",train_teacher_prefix_categories_one_hot
.shape)
print("Shape of cv data matrix after one hot encoding ",cv_teacher_prefix_categories_one_hot.shape
)
print("Shape of test data matrix after one hot encoding ",test_teacher_prefix_categories_one_hot.s
hape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of train data matrix after one hot encoding  (24520, 5)
Shape of train data matrix after one hot encoding  (12078, 5)
```

```
Shape of test data matrix after one hot encoding  (18026, 5)
```

In [85]:

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

X_train_price_std = standard_vec.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_std = standard_vec.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_std = standard_vec.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_price_std.shape, y_train.shape)
print(X_cv_price_std.shape, y_cv.shape)
print(X_test_price_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [89]:

```
X_train['price'].values
```

Out[89]:

```
array([  8.45,  89.45,   9.49, ..., 198.99,   6.  , 479.99])
```

In [90]:

```
X_train_price_std
```

Out[90]:

```
array([[0.02365447],
       [0.25040145],
       [0.02656579],
       ...,
       [0.55704175],
       [0.01679607],
       [1.34365782]])
```

In [91]:

```
#Normalising the numerical feature-no of words in essay
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['words_essay'].values.reshape(-1,1))

X_train_words_essay_std = standard_vec.transform(X_train['words_essay'].values.reshape(-1,1))
X_cv_words_essay_std = standard_vec.transform(X_cv['words_essay'].values.reshape(-1,1))
X_test_words_essay_std = standard_vec.transform(X_test['words_essay'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_words_essay_std.shape, y_train.shape)
print(X_cv_words_essay_std.shape, y_cv.shape)
print(X_test_words_essay_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [92]:

```
#Normalising the numerical feature-no of words in titles
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vec.fit(X_train['words_title'].values.reshape(-1,1))

X_train_words_title_std = standard_vec.transform(X_train['words_title'].values.reshape(-1,1))
X_cv_words_title_std = standard_vec.transform(X_cv['words_title'].values.reshape(-1,1))
X_test_words_title_std = standard_vec.transform(X_test['words_title'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_words_title_std.shape, y_train.shape)
print(X_cv_words_title_std.shape, y_cv.shape)
print(X_test_words_title_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [93]:

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

X_train_sentiment_neg_std = standard_vec.transform(X_train['sentiment_neg'].values.reshape(-1,1))
X_cv_sentiment_neg_std = standard_vec.transform(X_cv['sentiment_neg'].values.reshape(-1,1))
X_test_sentiment_neg_std = standard_vec.transform(X_test['sentiment_neg'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_sentiment_neg_std.shape, y_train.shape)
print(X_cv_sentiment_neg_std.shape, y_cv.shape)
print(X_test_sentiment_neg_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [94]:

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

X_train_sentiment_neu_std = standard_vec.transform(X_train['sentiment_neu'].values.reshape(-1,1))
```

```
X_cv_sentiment_neu_std = standard_vec.transform(X_cv['sentiment_neu'].values.reshape(-1,1))
X_test_sentiment_neu_std = standard_vec.transform(X_test['sentiment_neu'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_sentiment_neu_std.shape, y_train.shape)
print(X_cv_sentiment_neu_std.shape, y_cv.shape)
print(X_test_sentiment_neu_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [95]:

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

X_train_sentiment_pos_std = standard_vec.transform(X_train['sentiment_pos'].values.reshape(-1,1))
X_cv_sentiment_pos_std = standard_vec.transform(X_cv['sentiment_pos'].values.reshape(-1,1))
X_test_sentiment_pos_std = standard_vec.transform(X_test['sentiment_pos'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_sentiment_pos_std.shape, y_train.shape)
print(X_cv_sentiment_pos_std.shape, y_cv.shape)
print(X_test_sentiment_pos_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [96]:

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

X_train_sentiment_compound_std =
standard_vec.transform(X_train['sentiment_compound'].values.reshape(-1,1))
X_cv_sentiment_compound_std = standard_vec.transform(X_cv['sentiment_compound'].values.reshape(-1,
1))
X_test_sentiment_compound_std = standard_vec.transform(X_test['sentiment_compound'].values.reshape
(-1,1))

print("After vectorizations")
print(X_train_sentiment_compound_std.shape, y_train.shape)
print(X_cv_sentiment_compound_std.shape, y_cv.shape)
print(X_test_sentiment_compound_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [97]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
```

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

X_train_projects_std =
standard_vec.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1
))
X_cv_projects_std = standard_vec.transform(X_cv['teacher_number_of_previously_posted_projects'].va
lues.reshape(-1,1))
X_test_projects_std = standard_vec.transform(X_test['teacher_number_of_previously_posted_projects'
].values.reshape(-1,1))

print("After vectorizations")
print(X_train_projects_std.shape, y_train.shape)
print(X_cv_projects_std.shape, y_cv.shape)
print(X_test_projects_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

In [98]:

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

X_train_qty_std = standard_vec.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_qty_std = standard_vec.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_qty_std = standard_vec.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_qty_std.shape, y_train.shape)
print(X_cv_qty_std.shape, y_cv.shape)
print(X_test_qty_std.shape, y_test.shape)
```

```
After vectorizations
(24520, 1) (24520,)
(12078, 1) (12078,)
(18026, 1) (18026,)
```

2.4 Merge the features from step 3 and step 4

2.5 Apply XGBoost on the Final Features from the above section

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

In [105]:

```
# 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
```

```
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
# 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
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr =
hstack((train_essay_tfidf,X_train_sentiment_compound_std,X_train_sentiment_pos_std,X_train_sentimer
t_neu_std,X_train_sentiment_neg_std,X_train_words_title_std,X_train_words_essay_std,train_categorie
s_one_hot,train_subcategories_one_hot, train_school_state_category_one_hot,
train_teacher_prefix_categories_one_hot, train_project_grade_category_one_hot, X_train_price_std,X
_train_projects_std,X_train_qty_std)).tocsr()
X_cr =
hstack((cv_essay_tfidf,X_cv_sentiment_compound_std,X_cv_sentiment_pos_std,X_cv_sentiment_neu_std,X
_cv_sentiment_neg_std,X_cv_words_title_std,X_cv_words_essay_std,cv_categories_one_hot,cv_subcategor
ies_one_hot, cv_school_state_category_one_hot, cv_teacher_prefix_categories_one_hot,
cv_project_grade_category_one_hot, X_cv_price_std,X_cv_projects_std,X_cv_qty_std)).tocsr()
X_te =
hstack((test_essay_tfidf,X_test_sentiment_compound_std,X_test_sentiment_pos_std,X_test_sentiment_ne
u_std,X_test_sentiment_neg_std,X_test_words_title_std,X_test_words_essay_std,test_categories_one_ho
t,test_subcategories_one_hot, test_school_state_category_one_hot,
test_teacher_prefix_categories_one_hot, test_project_grade_category_one_hot, X_test_price_std,X_te
st_projects_std,X_test_qty_std)).tocsr()

print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(24520, 9548) (24520,)
(12078, 9548) (12078,)
(18026, 9548) (18026,)
====================================================================================================
```

In [112]:

```
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
tuned_parameters = {'max_depth': [2,4,6,8,10],
                    'n_estimators' : [50,100,150,200,250]}

base_estimator = XGBClassifier()


rsearch_cv = RandomizedSearchCV(base_estimator,param_distributions=tuned_parameters,cv=3, scoring='
roc_auc',n_jobs=-1)

rsearch_cv.fit(X_tr,y_train)
```

Out[112]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
          estimator=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=3, min_child_weight=1, missing=None,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1),
          fit_params=None, iid='warn', n_iter=10, n_jobs=-1,
          param_distributions={'max_depth': [2, 4, 6, 8, 10], 'n_estimators': [50, 100, 150, 200, 2
50]},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring='roc_auc', verbose=0)
```

In [113]:

```
print("Best estimator obtained from CV data: \n", rsearch_cv.best_estimator_)
print("Best Score : ", rsearch_cv.best_score_)
```

```
Best estimator obtained from CV data:
 XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=2, min_child_weight=1, missing=None,
       n_estimators=250, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
Best Score :  0.7337852228723539
```

In [115]:

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

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

    return y_data_pred
```

In [114]:

```
rsearch_cv.best_params_
```

Out[114]:

```
{'n_estimators': 250, 'max_depth': 2}
```

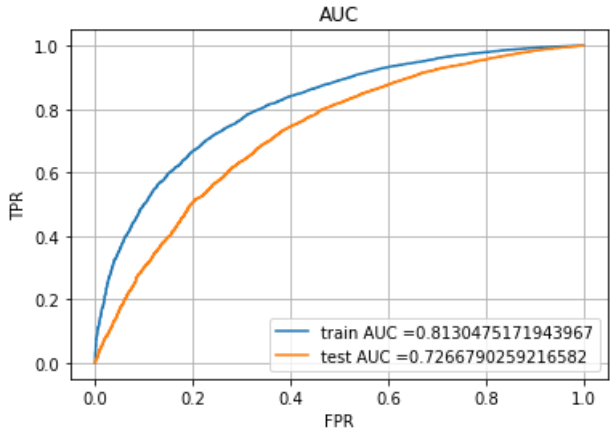
In [116]:

```
n_est = rsearch_cv.best_params_['n_estimators']
max_d = rsearch_cv.best_params_['max_depth']
```

In [118]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression
from sklearn.calibration import CalibratedClassifierCV
from sklearn import tree

clf = XGBClassifier(n_estimators=n_est,max_depth=max_d)
#https://github.com/scikit-learn/scikit-learn/issues/7278
# calibrated_clf = CalibratedClassifierCV(sgd, method='sigmoid')
clf.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(clf, X_tr)
y_test_pred = batch_predict(clf, X_te)

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

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

```
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



3. Conclusion

XGBoost seems to be the best implementation for Gradient Boost Decision Trees as compared to other implementations

This can be inferred from the test auc score and also the train time complexity as compared to other implementations of GBDT