DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

- Literature & Writing, Social Sciences

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

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

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 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]:

```
from sklearn.utils import resample
project_data=resample(project_data,n_samples=50000)
```

In [5]:

```
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[5]:

	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 [6]:

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

project_data['clean_subcategories'] = sub_cat_list
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 [9]:

```
# 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 [10]:

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

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	proj
0	23360	p115962	115331e4a8268e26151cbcf55b6f9e30	Ms.	CA	2016-11-06 14:02:10	Gra
1	162046	p059531	a1f80c84d1beadc2c4e57344cebb085e	Mrs.	IL	2016-08-16 23:30:24	Gra

	Unnamed:	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro

In [11]:

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

In [12]:

```
# 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)
```

```
Our school is in a unique mix of residential and high-tech industrial area. Many of our students a
re brand new residents of the United States, as their families have moved here for the opportunity
to work for some of these large tech companies. These students are coming into fourth grade with l
ittle to no English language knowledge and they give their absolute best every day to progress to
fluency and mastery of fourth grade concepts. It is an amazingly inspirational experience I get to
observe and help facilitate. \r\n\r\nIn essence, the wonderful community of learners I get to see
every day is comprised of members of our FUTURE. This idea and the perseverance I see in my studen
ts is what keeps me motivated to provide the best I possibly can for them. \r\n\r\n\r\nIn the past
, I've had my doubts about every student needing an iPad or a Chromebook. I feel that our society
relies too heavily on devices to avoid face-to-face interaction and replace the skills of
parenting, among other things. However, after attending many events and being in various arenas wh
ere technology in education is the focus, I've been exposed to the great potential for technology
to be an enhancement of education.\r\n\r\nIn fact, after these experiences, I believe that the
MOST important place for technology to be prevalently used is within the education system. It's tr
uthfully difficult to portray the plethora of beneficial experiences technology can bring into the
classroom but to list a few:\r\n\r\nQR codes: scannable barcodes that link to any pre-selected con
tent. Example of one use - (teacher cloning) teacher records him/herself teaching several differen
t lessons targeting same skill but with varying levels of difficulty. Students groups scan the bar
code and hear the lesson taught at a level that is accessible to them, all within the same class p
eriod.\r\n\r\nThingLink: students can virtually annotate a photo or video. Example of one use - (r
esearch and presentation skills) students research a topic, upload a photo of their work, then sel
ect various parts of the photo to attach links to \"cite\" their sources.\r\n\r\nMovieMaker: there
are many apps that allow students to create stop motion videos and do some \"post-production\" edi
ting. Example of one use - (demonstration of understanding) students learn about the rock cycle, t
hen use drawings or objects to make a video of the process, while narrating with a list of
important vocabulary that must be integrated.\r\n\r\nClassDojo: this is one of many apps that allo
w students to create online portfolios. Example of one use: digital portfolios to keep parents upd
ated on what we are producing.nannan
==================================================
Kindergarten students are adorable and like to show off everything they know.  1st grade students
try to contain their excitement about art but need lots of routine practice.  Second grade
students are attentive and willing to try anything, confident, and well behaved. Most Third grade
students are chatty, annoyed, and distracted by each other.  I do my best to engage with their int
erests and build their confidence. Fourth grade students are generally trying to be super cool and
pretend they already know everything; they need to be pushed a lot. Fifth grade students are great
because none of the them give up, they usually try their best and are resourceful and help each ot
her. I have two mixed grade special education classes I enjoy because they challenge me to adapt a
nd try new lessons. There is a wide range of personalities and skill levels in all of my
classes.My Students need to experience and use more than 8 colors, our class packs of crayons and
colored pencils provided by the school have a limited color palette - 8 colors . Our wold is not m
ade up of only 8 colors and my students need to see, and identify, and use  -  light brown, dark b
rown, sand,  peach, tan, as skin tones or neutral tones.  \r\n\r\n\r\nHaving a variety of color
choices gets students to identify and describe specific colors and color schemes such as warm colo
rs, cool colors, lights, darks, neutral, pastel, metallic.  Texture plates will be used to teach t
he element of texture and create textures in student artworks.  I will use the display easel for t
he top of my cart I use to teach 2 days a week.nannan
==================================================
My students are a diverse group of kiddos from East Oakland, California.  Many are learning Englis
h as a second language while soaking up all of the knowledge that first grade has to offer.
\r\nThese students come to school each day ready to challenge themselves and use their growth mind
set, despite the many struggles they may face at home or in their surrounding neighborhoods.  \r\n
We are a Title 1 School and 100% of my student population qualifies for free and reduced school lu
nch.  Many of our students have experienced trauma in and around their homes and neighborhoods and
our school emphasizes a strong culture of social emotional learning.  Our public charter school be
```

```
lieves that every student has the power to go out and change the world and we emphasize this in ou
r classrooms and school yard every day.  \r\nHaving a small group set of iPod Shuffles will allow
my students independent access to a variety of texts and in turn increase their reading ability, c
omprehension and vocabulary.  They will be able to listen to a variety of 1st grade appropriate te
xt that they may not have access to on their own or at home.  \r\nMany of my students are
struggling readers and providing them with the opportunity to listen to a favorite book or a story
without the aid of a teacher or classmate will allow them to be curious and excited about reading
without the anxiety of not being able to access the text.  \r\nWith the iPod shuffle as one of our
Guided Reading stations all students will be able to participate in reading a variety of grade lev
el texts each day and building their skills as confident readers.nannan
==================================================
I have an eager group of students who are so excited to come to school! Many of my students do not
attend preschool so kindergarten is their first school experience. Students come in with varying a
bilities from being able to read and solve math problems to knowing a few letters of the alphabet
and counting to 10. \r\n\r\nEvery day they come to school ready for more fun! They love hands-on a
ctivities to help them learn and practice new skills. I am hoping to be able to provide them with
more centers and independent activities to continue their love of learning.We have done guided rea
ding groups for years to meet the individual needs of ALL students. Now it is time to implement gu
ided math groups. Just like in reading, kids come to school with varying math abilities.
\r\n\r\nTo meet the needs of all of my students we need a variety of math games and activities tha
t students can complete independently while I am working with individual or small groups of
students. The items we are requesting will coincide with what the student(s) are being taught so t
hat they can feel successful while working on a math concept/skill. \r\n\r\nI am so excited to
implement guided math so that I can meet my students at their level and they can feel successful w
hile working independently or with a partner.nannan
==================================================
==================================================
```

In [13]:

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

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

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

In [14]:

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

```
I have an eager group of students who are so excited to come to school! Many of my students do not
attend preschool so kindergarten is their first school experience. Students come in with varying a
bilities from being able to read and solve math problems to knowing a few letters of the alphabet
and counting to 10. \r\n\r\nEvery day they come to school ready for more fun! They love hands-on a
ctivities to help them learn and practice new skills. I am hoping to be able to provide them with
more centers and independent activities to continue their love of learning.We have done guided rea
ding groups for years to meet the individual needs of ALL students. Now it is time to implement gu
ided math groups. Just like in reading, kids come to school with varying math abilities.
\r\n\r\nTo meet the needs of all of my students we need a variety of math games and activities tha
t students can complete independently while I am working with individual or small groups of
students. The items we are requesting will coincide with what the student(s) are being taught so t
hat they can feel successful while working on a math concept/skill. \r\n\r\nI am so excited to
implement guided math so that I can meet my students at their level and they can feel successful w
hile working independently or with a partner.nannan
==================================================
```

In [15]:

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

```
I have an eager group of students who are so excited to come to school! Many of my students do not
attend preschool so kindergarten is their first school experience. Students come in with varying a
bilities from being able to read and solve math problems to knowing a few letters of the alphabet
and counting to 10.     Every day they come to school ready for more fun! They love hands-on
activities to help them learn and practice new skills. I am hoping to be able to provide them with
more centers and independent activities to continue their love of learning.We have done guided rea
ding groups for years to meet the individual needs of ALL students. Now it is time to implement gu
ided math groups. Just like in reading, kids come to school with varying math abilities.     To
meet the needs of all of my students we need a variety of math games and activities that students
can complete independently while I am working with individual or small groups of students. The
items we are requesting will coincide with what the student(s) are being taught so that they can f
eel successful while working on a math concept/skill.     I am so excited to implement guided math
so that I can meet my students at their level and they can feel successful while working
independently or with a partner.nannan
```

In [16]:

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

```
I have an eager group of students who are so excited to come to school Many of my students do not
attend preschool so kindergarten is their first school experience Students come in with varying
abilities from being able to read and solve math problems to knowing a few letters of the alphabet
and counting to 10 Every day they come to school ready for more fun They love hands on activities
to help them learn and practice new skills I am hoping to be able to provide them with more center
s and independent activities to continue their love of learning We have done guided reading groups
for years to meet the individual needs of ALL students Now it is time to implement guided math gro
ups Just like in reading kids come to school with varying math abilities To meet the needs of all
of my students we need a variety of math games and activities that students can complete
independently while I am working with individual or small groups of students The items we are
requesting will coincide with what the student s are being taught so that they can feel successful
while working on a math concept skill I am so excited to implement guided math so that I can meet
my students at their level and they can feel successful while working independently or with a
partner nannan
```

In [17]:

```
# 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', "do
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 [18]:

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

```
100%|███████████████████████████████████████████████████████████████████████████| 50000/50000
[00:42<00:00, 1169.66it/s]
```

In [19]:

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

Out[19]:

```
'i eager group students excited come school many students not attend preschool kindergarten first
school experience students come varying abilities able read solve math problems knowing letters al
phabet counting 10 every day come school ready fun they love hands activities help learn practice
new skills i hoping able provide centers independent activities continue love learning we done gui
ded reading groups years meet individual needs all students now time implement guided math groups
just like reading kids come school varying math abilities to meet needs students need variety math
games activities students complete independently i working individual small groups students the it
ems requesting coincide student taught feel successful working math concept skill i excited
implement guided math i meet students level feel successful working independently partner nannan'
```

In [20]:

```
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 [21]:

```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_title= []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].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_title.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:01<00:00, 30736.79it/s]
```

In [23]:

```
project_data['clean_title'] = preprocessed_title
```

```
project_data['clean_title'] = preprocessed_title
```

In [27]:

```
y = project_data['project_is_approved'].values

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

X = project_data
```

In [28]:

```
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 [29]:

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

print("="*100)
```

```
(22445, 17) (22445,)
(11055, 17) (11055,)
(16500, 17) (16500,)
====================================================================================================
```

In [30]:

```
i="Divya is nan   a   good         girl"
list_i = i.replace('^nan',' ')
print(list_i)
```

```
Divya is nan   a   good         girl
```

Checking for Sample data

In [31]:

```
list_word = ["ABC","PQR","DEF"]
combined_essay = ["ABC DEF IJK PQR", "PQR KLM OPQ", "LMN PQR XYZ ABC DEF PQR ABC"]
```

In [32]:

```
arr =[]
for i in combined_essay:
    arr .append (i.split())
```

In [33]:

```
arr
```

Out[33]:

```
[['ABC', 'DEF', 'IJK', 'PQR'],
 ['PQR', 'KLM', 'OPQ'],
 ['LMN', 'PQR', 'XYZ', 'ABC', 'DEF', 'PQR', 'ABC']]
```

In [34]:

```
co_variance = pd. DataFrame (np.zeros(9).reshape(3,3))
co_variance .columns = list_word
co_variance . index = list_word
```

In [35]:

```
co_variance
```

Out[35]:

	ABC	PQR	DEF
ABC	0.0	0.0	0.0
PQR	0.0	0.0	0.0
DEF	0.0	0.0	0.0

In [36]:

```
%%time
for i,w in enumerate(list_word):
    #print(w,"word")
    for line in arr:
        #print(line,"line")

        if w in line:
            list_search = list_word[i+1:3]
            #print(list_search,"list_search")
            #print("It is in list")
            for search in list_search:
                #print(search)
                if search in line:
                    count = 0
                    flag = 0
                    indices_search = [i for i, x in enumerate(line) if x == search]
                    indices_word = [i for i, x in enumerate(line) if x == w]
                    #print(indices_search , indices_word)
                    for j in indices_search:
                        for k in indices_word:
                            if abs(j-k)<=2:
                                count += 1
                                flag=1

                    if flag==1:
                        co_variance .loc[w ,search] += count
                        co_variance.loc[search ,w] += count
                    #print(co_variance)
```

```
Wall time: 0 ns
```

In [37]:

```
co_variance
```

Out[37]:

	ABC	PQR	DEF
ABC	0.0	3.0	3.0
PQR	3.0	0.0	2.0
DEF	3.0	2.0	0.0

In [38]:

```
print(len(preprocessed_essays))
N =(len(preprocessed_title))
```

```
50000
```

In [39]:

```
DF = {}

for i in range(N):
    tokens = preprocessed_essays[i]
    for w in tokens:
        try:
            DF[w].add(i)
        except:
            DF[w] = {i}

    tokens = preprocessed_title[i]
    for w in tokens:
        try:
            DF[w].add(i)
        except:
            DF[w] = {i}
for i in DF:
    DF[i] = len(DF[i])
```

In [40]:

```
train_essays = X_train['clean_essays']
train_title = X_train['clean_title']
test_essays = X_test['clean_essays']
test_title = X_test['clean_title']
cv_essays = X_cv['clean_essays']
cv_title = X_cv['clean_title']
```

In [41]:

```
train_essays.iloc[100].find("nan")
```

Out[41]:

```
909
```

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

In [42]:

```
train_essaycombined = train_essays  + train_title
test_essaycombined = test_essays  + test_title
cv_essaycombined = cv_essays  + cv_title
```

In [45]:

```
%%time
# reference https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-r
eal-world-dataset-796d339a4089


def word_vector(processed_combined):
    DF = {}
    for i,w in enumerate(processed_combined):
        list_words = w.split()
        #print(len(list_words))
        tokens = list(set(list_words))
        #print(tokens)
        for j in tokens:
            #print(j)
            try:
                DF[j].add(i)
            except:
                DF[j] = {i}
    idf = {}
    for i in DF:
        DF[i] = len(DF[i])
        df = DF[i]
        idf[i] = np.log((N+1)/(df+1))
    #print((idf))
    print(len(idf))
    dictionary_idf =sorted(idf.items(),reverse=True, key=lambda item: item[1])
    dictionary_idf = dictionary_idf[:2000]
    list_word =[]
    for i in dictionary_idf:
        list_word .append(i[0])
    print(len(list_word))
    return list_word

#print(DF)
```

```
Wall time: 0 ns
```

In [46]:

```
train_essaycombined = [re.sub(r"nan" , "",i) for i in train_essaycombined ]
cv_essaycombined = [re.sub(r"nan" , "",i) for i in cv_essaycombined ]
test_essaycombined = [re.sub(r"nan" , "",i) for i in test_essaycombined ]
```

In [47]:

```
test_essaycombined[100].find("nan")
```

Out[47]:

```
-1
```

In [48]:

```
trainessays_words= word_vector(train_essaycombined)
testessays_words= word_vector(test_essaycombined)
cvessays_words= word_vector(cv_essaycombined)
```

```
31180
2000
28004
2000
23706
2000
```

In [49]:

```
# removing nan from words
trainessays_words = [re.sub(r"nan" , "",i) for i in trainessays_words ]
cvessays_words = [re.sub(r"nan" , "",i) for i in cvessays_words ]
testessays_words = [re.sub(r"nan" , "",i) for i in testessays_words ]
```

In [50]:

```
trainessays_words[:10]
```

Out[50]:

```
['northridge',
 'guin',
 'distractive',
 'skipper',
 'ballance',
 'emboldens',
 'absorption',
 'enlish',
 'haitain',
 'mementos']
```

2.2 Computing Co-occurance matrix

In [51]:

```
def create_covariance(list_word ,essaycombined):
    co_variance = pd. DataFrame (np.zeros(4000000).reshape(2000,2000))
    co_variance .columns = list_word
    co_variance . index = list_word
    arr =[]
    for i in essaycombined:
        arr .append (i.split())
    for i,w in enumerate(list_word):
        for line in arr:
            if w in line:
                list_search = list_word[i+1:2000]
                for search in list_search:
                    if search in line:
                        count = 0
                        flag = 0
                        indices_search = [i for i, x in enumerate(line) if x == search]
                        indices_word = [i for i, x in enumerate(line) if x == w]
                        #print(indices_search , indices_word)
                        for j in indices_search:
                            for k in indices_word:
                                if abs(j-k)<=5:
                                    count += 1
                                    flag=1

                        if flag==1:
                            co_variance .loc[w ,search] += count
                            co_variance.loc[search ,w] += count
    return co_variance
```

In [52]:

```
%%time
X_train_essays = create_covariance(trainessays_words , train_essaycombined)
X_cv_essays = create_covariance(cvessays_words , cv_essaycombined)
X_test_essays = create_covariance(testessays_words , test_essaycombined)
```

```
Wall time: 4min 13s
```

Truncated SVD matrix

In [53]:

```
from sklearn.decomposition import TruncatedSVD
from scipy import sparse
variance_list=[]
list_i=[10,100,200,300,500,750,1000,1300,1500,1750]
variance_list_test=[]
for i in (list_i):
    svd = TruncatedSVD(n_components=i, n_iter=3, random_state=42)
    svd.fit(sparse.csr_matrix(X_train_essays) )
    variance_list .append(svd.explained_variance_.sum())
    print(i,svd.explained_variance_.sum())
for i in (list_i):
    svd = TruncatedSVD(n_components=i, n_iter=3, random_state=42)
    svd.fit(sparse.csr_matrix(X_test_essays) )
    variance_list_test .append(svd.explained_variance_.sum())
    print(i,svd.explained_variance_.sum())
```

```
10 0.13666123135623723
100 0.2917780274606508
200 0.3428743916414434
300 0.3929614033742139
500 0.4365644999999998
750 0.43656450000000013
1000 0.4365645000000003
1300 0.43656450000000074
1500 0.4365644999999981
```

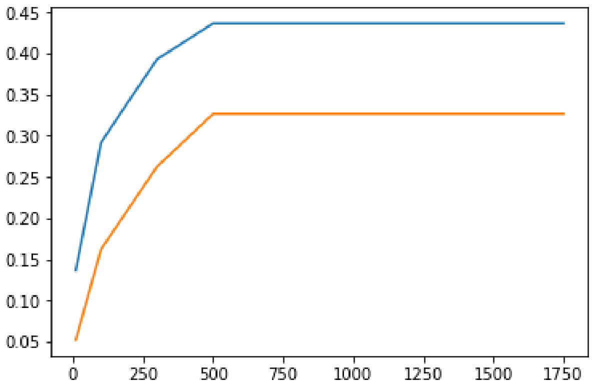
```
1750 0.4365645000000017
10 0.0521529563476717
100 0.1622770658693448
200 0.21253685110080944
300 0.2625736715045041
500 0.3267585
750 0.3267584999999994
1000 0.32675850000000006
1300 0.3267585000000008
1500 0.32675849999999923
1750 0.3267584999999996
```

In [54]:

```
plt.plot(list_i,variance_list)
plt.plot(list_i,variance_list_test)
```

Out[54]:

```
[<matplotlib.lines.Line2D at 0x2a6f96c6da0>]
```



In [55]:

```
svd = TruncatedSVD(n_components=500, n_iter=3, random_state=18)
X_train_essays_SVD=svd.fit_transform(sparse.csr_matrix(X_train_essays))
X_test_essays_SVD=svd.fit_transform(sparse.csr_matrix(X_test_essays))
```

In [56]:

```
type(X_train_essays_SVD)
```

Out[56]:

```
numpy.ndarray
```

In [57]:

```
X_test_essays_SVD.shape
```

Out[57]:

```
(2000, 500)
```

In [58]:

```
list_holding = []
for i in train_essaycombined:
    list_holding .append( len(i.split()))
X_train['word_count'] = list_holding

list_holding = []
for i in test_essaycombined:
    list_holding .append( len(i.split()))
X_test['word_count'] = list_holding

list_holding = []
list_holding = []
for i in cv_essaycombined:
    list_holding .append( len(i.split()))
X_cv['word_count'] = list_holding

list_holding = []
for i in train_title:
    list_holding .append( len(i.split()))
X_train['title_count'] = list_holding

list_holding = []
for i in test_title:
    list_holding .append( len(i.split()))
X_test['title_count'] = list_holding

list_holding = []
for i in cv_title:
    list_holding .append( len(i.split()))
X_cv['title_count'] = list_holding
```

In [59]:

```
print((X_train['word_count'])[:100])
print((X_train['title_count'][:100]))
```

```
42636    116
19749    147
17400    149
43843    192
15202    178
29114    150
17774    126
46556    190
21883    137
33785    126
41005    111
47845    150
28380    138
29334    119
11711    190
6220     129
40424    126
3939     107
40173    145
49046    246
21188    183
7867     242
18012    143
36102    123
23345    146
45308    101
30942    145
44096    200
10916    224
5398     136
        ...
7067     155
43154    215
7378     106
20137    147
40389    126
9532     128
34222    149
30302    135
46708    168
30193    240
738      173
47614    195
44740    116
1468     102
32299    113
30608    132
28503    143
9117     142
1689     241
```

```
...
14699    256
8444     138
37105    136
11551    166
13322    223
14218    209
40192    168
42440    220
37366    118
15465    200
31227    177
Name: word_count, Length: 100, dtype: int64
42636     3
19749     9
17400     5
43843    10
15202    10
29114     4
17774     3
46556     6
21883     3
33785     5
41005     3
47845     5
28380     2
29334     4
11711     4
6220      5
40424     7
3939      2
40173     3
49046     8
21188     5
7867      6
18012     4
36102     4
23345     3
45308     2
30942     4
44096     6
10916     5
5398      5
         ..
7067      4
43154     5
7378      3
20137     5
40389     7
9532      4
34222     2
30302     7
46708     5
30193     5
738       7
47614     3
44740     3
1468      3
32299     3
30608     4
28503     5
9117      4
1689      4
14699    10
8444      3
37105     5
11551     5
13322     5
14218     5
40192     5
42440     6
37366     9
15465     4
31227     4
Name: title_count, Length: 100, dtype: int64
```

In [60]:

In [60]:

```
project_data.columns
```

Out[60]:

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

In [61]:

```
from sklearn.preprocessing import OneHotEncoder,LabelEncoder
def OneHotEncoder_find(category):
    le = LabelEncoder()
    enc = OneHotEncoder()
    train_ohe = enc.fit_transform(le.fit_transform(X_train[category].astype(str).values).reshape(-1
, 1))
    test_ohe = enc.fit_transform(le.fit_transform(X_test[category].astype(str).values).reshape(-1,
1))
    #train_ohe = enc.fit_transform(X_train[category])
    #test_ohe = enc.fit_transform(X_test[category])
    return train_ohe ,test_ohe
```

In [62]:

```
X_train_clean_cat,X_te_clean_cat = OneHotEncoder_find('clean_categories')
X_train_clean_subcat,X_te_clean_subcat = OneHotEncoder_find('clean_subcategories')
X_train_grade ,X_test_grade  = OneHotEncoder_find('project_grade_category')
X_train_teacher,X_test_teacher = OneHotEncoder_find('teacher_prefix')
X_train_state,X_test_state = OneHotEncoder_find('school_state')
```

In [63]:

```
print((X_train_clean_cat).shape)
print(X_te_clean_cat.shape)
print(X_test.shape)
```

```
(22445, 49)
(16500, 47)
(16500, 19)
```

In [75]:

```
essays =X_train['clean_essays']
essays = essays[:2000]
print(type(essays))
X_train = X_train[:2000]
X_test = X_test[:2000]
```

```
<class 'pandas.core.series.Series'>
```

Computing Sentiment Scores

In [94]:

```
%%time
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
list_neg =[]
list_neu =[]
list_pos = []
list_comp =[]
essays = X_train['clean_essays']
for i in essays[:2000]:

    sid = SentimentIntensityAnalyzer()
    ss = sid.polarity_scores(i)

    for k in ss:
    #print('{0}: {1}, '.format(k, ss[k]), end='')
    #neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975
        if k=='neg':
            list_neg .append(ss[k])
        elif k=='neu':
            list_neu .append(ss[k])
        elif k=='pos':
            list_pos .append(ss[k])
        elif k=='compound':
            list_comp .append(ss[k])
X_train['sent_neg'] = list_neg
X_train['sent_neu'] = list_neu
X_train['sent_pos'] = list_pos
X_train['sent_comp'] = list_comp


list_neg =[]
list_neu =[]
list_pos = []
list_comp =[]
essays = X_test['clean_essays']
for i in essays[:2000]:

    sid = SentimentIntensityAnalyzer()
    ss = sid.polarity_scores(i)

    for k in ss:
    #print('{0}: {1}, '.format(k, ss[k]), end='')
    #neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975
        if k=='neg':
            list_neg .append(ss[k])
        elif k=='neu':
            list_neu .append(ss[k])
        elif k=='pos':
            list_pos .append(ss[k])
        elif k=='compound':
            list_comp .append(ss[k])
X_test['sent_neg'] = list_neg
X_test['sent_neu'] = list_neu
X_test['sent_pos'] = list_pos
X_test['sent_comp'] = list_comp
```

```
Wall time: 51.3 s
```

In [86]:

```
X_train.columns
```

Out[86]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'clean_categories', 'clean_subcategories', 'essay', 'clean_essays',
       'clean_title', 'word_count', 'title_count', 'sent_neg', 'sent_neu',
       'sent_pos', 'sent_comp'],
      dtype='object')
```

In [104]:

```
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)
print("="*100)
```

```
After vectorizations
(2000, 1) (22445,)
(11055, 1) (11055,)
(2000, 1) (16500,)
====================================================================================================
```

In [105]:

```
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['word_count'].values.reshape(-1,1))

X_train_word_std = standard_vec.transform(X_train['word_count'].values.reshape(-1,1))
X_cv_word_std = standard_vec.transform(X_cv['word_count'].values.reshape(-1,1))
X_test_word_std = standard_vec.transform(X_test['word_count'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_word_std.shape, y_train.shape)
print(X_cv_word_std.shape, y_cv.shape)
print(X_test_word_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(11055, 1) (11055,)
(2000, 1) (16500,)
```

In [106]:

```
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['word_count'].values.reshape(-1,1))

X_train_title_std = standard_vec.transform(X_train['title_count'].values.reshape(-1,1))
X_cv_title_std = standard_vec.transform(X_cv['title_count'].values.reshape(-1,1))
X_test_title_std = standard_vec.transform(X_test['title_count'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_title_std.shape, y_train.shape)
print(X_cv_title_std.shape, y_cv.shape)
print(X_test_title_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(11055, 1) (11055,)
(2000, 1) (16500,)
```

In [95]:

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

standard_vec.fit(X_train['sent_neg'].values.reshape(-1,1))
#print(X_train['sent_neg'])
```

```
#print(X_train['sent_neg'])

X_train_sent_neg_std = standard_vec.transform(X_train['sent_neg'].values.reshape(-1,1))
#X_cv_title_std = standard_vec.transform(X_cv['sent_neg'].values.reshape(-1,1))
X_test_sent_neg_std = standard_vec.transform(X_test['sent_neg'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_sent_neg_std.shape, y_train.shape)
#print(X_cv_title_std.shape, y_cv.shape)
print(X_test_sent_neg_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(2000, 1) (16500,)
```

In [96]:

```
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['sent_neu'].values.reshape(-1,1))

X_train_sent_neu_std = standard_vec.transform(X_train['sent_neu'].values.reshape(-1,1))
#X_cv_title_std = standard_vec.transform(X_cv['sent_neg'].values.reshape(-1,1))
X_test_sent_neu_std = standard_vec.transform(X_test['sent_neu'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_sent_neu_std.shape, y_train.shape)
#print(X_cv_title_std.shape, y_cv.shape)
print(X_test_sent_neu_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(2000, 1) (16500,)
```

In [97]:

```
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['sent_pos'].values.reshape(-1,1))

X_train_sent_pos_std = standard_vec.transform(X_train['sent_pos'].values.reshape(-1,1))
#X_cv_title_std = standard_vec.transform(X_cv['sent_neg'].values.reshape(-1,1))
X_test_sent_pos_std = standard_vec.transform(X_test['sent_pos'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_sent_pos_std.shape, y_train.shape)
#print(X_cv_title_std.shape, y_cv.shape)
print(X_test_sent_pos_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(2000, 1) (16500,)
```

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['sent_comp'].values.reshape(-1,1))
```

```
X_train_sent_comp_std = standard_vec.transform(X_train['sent_comp'].values.reshape(-1,1))
#X_cv_title_std = standard_vec.transform(X_cv['sent_neg'].values.reshape(-1,1))
X_test_sent_comp_std = standard_vec.transform(X_test['sent_comp'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_sent_comp_std.shape, y_train.shape)
#print(X_cv_title_std.shape, y_cv.shape)
print(X_test_sent_comp_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(2000, 1) (16500,)
```

In [107]:

```
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['price'].values.reshape(-1,1))

X_train_price_std = standard_vec.transform(X_train['price'].values.reshape(-1,1))
#X_cv_title_std = standard_vec.transform(X_cv['sent_neg'].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_title_std.shape, y_cv.shape)
print(X_test_price_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(2000, 1) (16500,)
```

In [113]:

```
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['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_title_std = standard_vec.transform(X_cv['sent_neg'].values.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_title_std.shape, y_cv.shape)
print(X_test_projects_std.shape, y_test.shape)
```

```
After vectorizations
(2000, 1) (22445,)
(2000, 1) (16500,)
```

In [114]:

```
X_test_essays_SVD .shape
```

Out[114]:

```
(2000, 500)
```

In [115]:

```
X_train = X_train[:2000]
X_test = X_test[:2000]
```

In [116]:

```
X_train_essays_SVD
```

Out[116]:

```
array([[-3.33887652e-19,  4.44503951e-16,  5.67552032e-18, ...,
         1.80933192e-19, -1.06062521e-19, -8.59491669e-20],
       [-3.03959029e-19,  3.41863380e-19,  4.63179042e-18, ...,
        -2.27334821e-19,  2.32524610e-19,  1.14034634e-20],
       [-3.60316517e-17, -4.00849435e-17,  5.87449950e-16, ...,
        -1.38302694e-18,  5.69065557e-18, -2.30140231e-18],
       ...,
       [ 0.00000000e+00,  0.00000000e+00, -0.00000000e+00, ...,
        -0.00000000e+00,  0.00000000e+00, -0.00000000e+00],
       [ 2.89290703e-19,  2.31645313e-18,  2.76602866e-17, ...,
        -1.44411938e-33, -8.30367625e-34,  1.57348462e-33],
       [ 1.62935225e-18,  9.69692698e-17, -4.59452045e-17, ...,
        -5.70982244e-34, -6.85356917e-34,  2.09024467e-33]])
```

In [117]:

```
X_test_essays_SVD
```

Out[117]:

```
array([[-4.46884382e-16,  1.28697792e-16,  7.64898328e-17, ...,
        -2.78663516e-33,  1.33079768e-33,  2.29972696e-33],
       [-2.21567249e-15, -3.26383705e-16,  2.39813310e-16, ...,
         2.36635427e-33,  4.67074814e-34,  1.56854383e-33],
       [ 2.21020774e-16, -7.51967564e-17, -4.16496121e-16, ...,
         8.42614380e-18, -9.43694108e-19,  1.02335159e-18],
       ...,
       [ 6.86393716e-17, -9.98540081e-17,  4.04165417e-17, ...,
         1.24315853e-33,  8.72804757e-35, -1.90206103e-33],
       [-0.00000000e+00, -0.00000000e+00, -0.00000000e+00, ...,
        -0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 1.88488661e-16,  7.26874525e-17, -1.18725103e-16, ...,
         3.66197206e-33,  2.15256608e-33, -8.84373585e-34]])
```

Concatinating all features

In [154]:

```
X_train_clean_cat = (X_train_clean_cat[:2000,:47])
X_train_clean_subcat = X_train_clean_subcat[:2000,:319]
X_train_teacher = X_train_teacher[:2000,:5]
```

In [155]:

```
from scipy.sparse import hstack
from scipy.sparse import hstack
X_tr = hstack((X_train_essays_SVD,X_train_clean_cat[:2000,:],X_train_clean_subcat[:2000,:], X_trai
n_state[:2000,:], X_train_teacher[:2000,:], X_train_grade[:2000,:], X_train_price_std[:2000,:],X_t
rain_projects_std[:2000,:],X_train_qty_std[:2000,:],X_train_word_std[:2000,:],X_train_title_std[:2
000,:],X_train_sent_comp_std,X_train_sent_pos_std,X_train_sent_neu_std,X_train_sent_neg_std)).tocs
r()
#X_cr =
hstack((X_cv_essays_SVD,X_cv_titles_tfidf,X_cv_summary_tfidf,X_cv_clean_cat_ohe,X_cv_clean_subcat_
X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe,
X_cv_price_std,X_cv_projects_std,X_cv_qty_std)).tocsr()
X_te = hstack((X_test_essays_SVD,X_te_clean_cat[:2000,:],X_te_clean_subcat[:2000,:], X_test_state[
:2000,:], X_test_teacher[:2000,:], X_test_grade[:2000,:], X_test_price_std[:2000,:],X_test_project
s_std[:2000,:],X_test_qty_std[:2000,:],X_test_word_std[:2000,:],X_test_title_std[:2000,:],X_test_s
ent_comp_std,X_test_sent_pos_std,X_test_sent_neu_std,X_test_sent_neg_std)).tocsr()

y_train = y_train[:2000]
y_test = y_test[:2000]
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
(2000, 935) (2000,)
(2000, 935) (2000,)
====================================================================================================
```

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**

2.5 Apply XGBoost on the Final Features from the above section

In [157]:

```
%%time
import warnings
warnings.filterwarnings("ignore")
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
import math
import xgboost as xgb
from sklearn.metrics import roc_auc_score
from sklearn import metrics
from sklearn import cross_validation
```

```
Wall time: 0 ns
```

In [158]:

```
X_tr = X_tr.toarray()
X_te = X_te.toarray()
```

In [159]:

```
type(y_train)
```

Out[159]:

```
numpy.ndarray
```

In [163]:

```
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, 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
```

In [185]:

```
%%time
clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4,)
##################################################################
#                Change from here                                #
##################################################################
```

```
parameters = {
    #'eta': [0.05,0.6,0.8, 0.1,0.2,0.5,.4, 0.3],
    'learning_rate': [0.1, 0.8, 0.4, 0.5 ,0.6,0.2],
    #'num_boost_round': [100, 250, 500],
    #'max_depth': [3, 5 ,6, 9, 12],
     'subsample': [0.1,0.5,0.7,0.9, 1.0],
}

clf = GridSearchCV(clf, parameters,cv=3, scoring='roc_auc')
#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_tr, y_train)

# 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]))
```

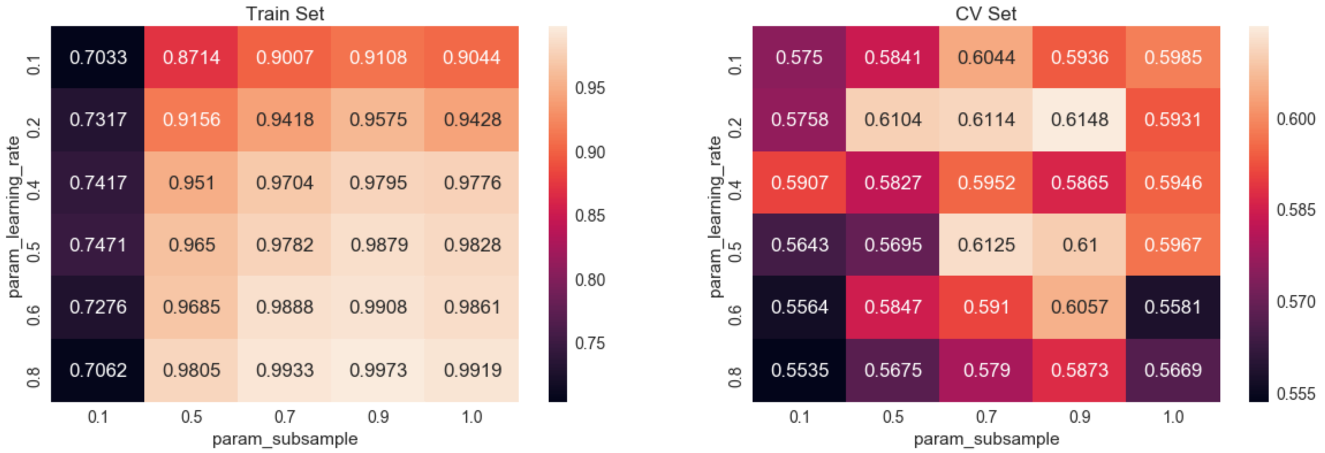
```
score: 0.6148373470648865
learning_rate: 0.2
subsample: 0.9
Wall time: 2min 14s
```

HeatMap using Seaborn

In [186]:

```
import seaborn as sns
import pandas as pd
#reference https://seaborn.pydata.org/generated/seaborn.heatmap.html ,
https://blog.exploratory.io/quick-introduction-to-heatmap-c21a9f9e4644?gi=cbae67554962

max_scores1 = pd.DataFrame(clf.cv_results_).groupby(['param_learning_rate',
'param_subsample']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



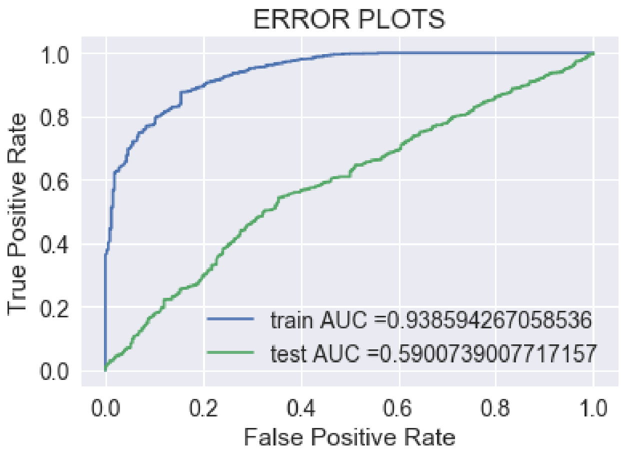
XGBoost with Optimal Parameters

In [182]:

```
from sklearn.model_selection import GridSearchCV
clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4)
best_parameters= {'learning_rate':[0.2],'subsample' :[0.9]}
clf = GridSearchCV(clf, best_parameters,cv=3, scoring='roc_auc')
clf.fit(X_tr, y_train)
```

```
print(clf.predict_proba(X_tr)[:,1])
y_train_pred= clf.predict_proba(X_tr)[:,1]
y_test_pred= clf.predict_proba(X_te)[:,1]
#print(X_tr , X_te)
train_fpr1, train_tpr1, tr_thresholds1 = roc_curve(y_train, y_train_pred)
test_fpr1, test_tpr1, te_thresholds1 = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr1, train_tpr1)))
plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr1, test_tpr1)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```

```
[0.7337475  0.8135566  0.3237442  ... 0.794764   0.88391733 0.89353365]
```



Confusion Matrix for test and train data

In [173]:

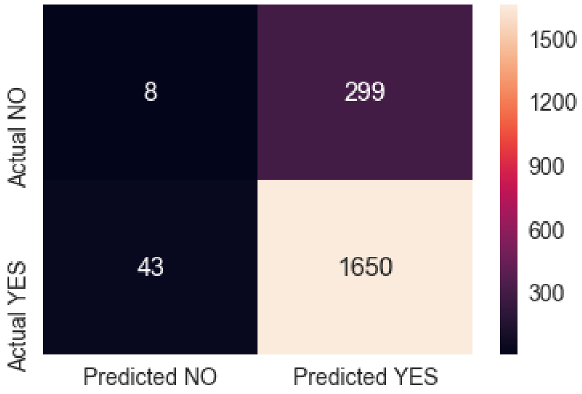
```
def get_confusion_matrix(y_test,y_pred):
    df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred), range(2),range(2))
    df_cm.columns = ['Predicted NO','Predicted YES']
    df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
    sns.set(font_scale=1.4)#for label size
    sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

In [176]:

```
predbow = (clf.predict(X_te))
predbow_train=(clf.predict(X_tr))
```

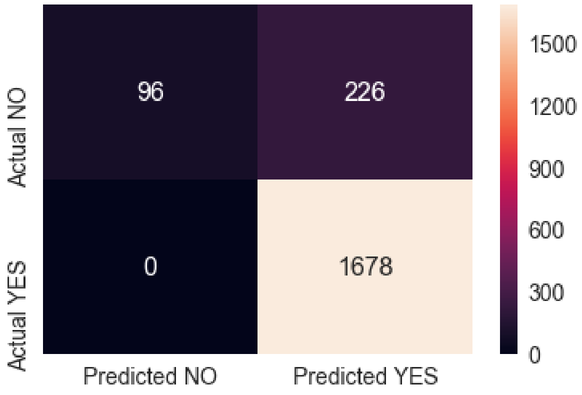
In [177]:

```
get_confusion_matrix(y_test,predbow)
```



In [178]:

```
get_confusion_matrix(y_train,predbow_train)
```



In [179]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test ,predbow))
```

```
             precision    recall  f1-score   support

          0       0.16      0.03      0.04       307
          1       0.85      0.97      0.91      1693

avg / total       0.74      0.83      0.77      2000
```

3. Conclusion

Observations:

1. For creating top features depending on IDF value , instead of using existing techniques likes TFIDF,AVG W2V , function which calculates IDF vaues of each word in al documnets was created
2. Creation of covariance matrix is done using pandas dataframe , as it would be easier to assign row and column names as top 2000 features.
3.Inorder to find index of each word index() was first used and observed that it returned index of only first occurence of word.Then another function was used to calcuate a indices of word.Inorder to find word in forward and backward direction abs() is used so that no additiona checks are required at boundaries.
4.Applying truncated SVD was easier and took less time and found that input for this shoud be sparse matrix.
5.Combining features in both train and test also was easy but there were some issues in bringing all data to same shape while applying hstack()
6.Atlast applying XGboost function took very less time and it was very fast compared to other algorithms.Two parameters were considered 'max_depth' and 'eta'.