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 (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **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!</code`
`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`

Feature	Description
`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 [3]:

```python
%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 [4]:

```python
project_data = pd.read_csv('train_data.csv', nrows = 50000)
resource_data = pd.read_csv('resources.csv')
```

In [5]:

```python
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 (50000, 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 [6]:

```python
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[6]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of project_subject_categories

In [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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Mat
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removin
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&
        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"=> "Mat
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removin
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&
        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

Removing null values from project essay 3 & 4

In [9]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_3'].isnull().values.any())
print("number of nan values",project_data['project_essay_3'].isnull().values.sum())
```

```
True
number of nan values 48315
```

In [10]:
```
#Replacing the Nan values with most frequent value in the column
project_data['project_essay_3']=project_data['project_essay_3'].fillna(' ')
```

In [11]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_3'].isnull().values.any())
print("number of nan values",project_data['project_essay_3'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [12]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_4'].isnull().values.any())
print("number of nan values",project_data['project_essay_4'].isnull().values.sum())
```

```
True
number of nan values 48315
```

In [13]:
```
#Replacing the Nan values with most frequent value in the column
project_data['project_essay_4']=project_data['project_essay_4'].fillna(' ')
```

In [14]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_4'].isnull().values.any())
print("number of nan values",project_data['project_essay_4'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [15]:
```
# 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 [16]:
```
project_data.head(2)
```

Out[16]:

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

1.4.2.3 Using Pretrained Models: TFIDF weighted W2V

In [17]:

```python
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[50])
print("="*50)
print(project_data['essay'].values[100])
print("="*50)
print(project_data['essay'].values[200])
print("="*50)
print(project_data['essay'].values[999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. We are a
melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our schoo
l. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every l
evel of mastery.  We also have over 40 countries represented with the families within our school.  Each s
tudent brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, an
d respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Our English
learner's have a strong support system at home that begs for more resources.  Many times our parents are
learning to read and speak English along side of their children.  Sometimes this creates barriers for par
ents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r
\nBy providing these dvd's and players, students are able to continue their mastery of the English langua
ge even if no one at home is able to assist.  All families with students within the Level 1 proficiency s
tatus, will be a offered to be a part of this program.  These educational videos will be specially chosen
by the English Learner Teacher and will be sent home regularly to watch.  The videos are to help the chil
d develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the oppo
rtunity to check out a dvd player to use for the year.  The plan is to use these videos and educational d
vd's for the years to come for other EL students.\r\n
==================================================
The students in our rural NC school come from various backgrounds with many different learning styles and
abilities. Many are from military families that have a mother or a father that are deployed. A large port
```

In [18]:

```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 [19]:

```python
sent = decontracted(project_data['essay'].values[200])
print(sent)
print("="*50)
```

```
As an inclusion kindergarten teacher, I am constantly looking for materials to help students develop and gr
ow throughout the school year.  This has been challenging with the school is limited funding for supplies.
\r\n\r\nWe are a classroom of 20 friendly and curious learners, from various ethnic backgrounds, facing cha
llenges, including poverty and developmental delays.\r\n\r\nMy students are future scholars, teachers, doct
ors, and accomplished human beings.  I need the public is help to raise money for materials that help maint
ain the attention of my special needs students.Last year was my first year teaching Kindergarten inclusion.
I learned that students can wiggle, and learn at the same time!\r\n\r\nMy students need sensory toys to mai
ntain focus on simple tasks that will shape their social and academic future.  My students with ADHD find t
hemselves moving their hands, feet, and bodies without much control.  Sensory toys help my students use the
ir energy in a positive manner (fidget toys, bouncy chairs, etc).  With fidget toys, my students use their
energy to play appropriately while listening at the same time.\r\n\r\nI have noticed that my students with
special needs are able to pay attention when they are given the proper tools and models to succeed.  My goa
l is to accommodate young learners with special needs, and allow them to express themselves in a positive w
ay that will lead to their success now and in the future.
==================================================
```

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)
```

```
As an inclusion kindergarten teacher, I am constantly looking for materials to help students develop and gr
ow throughout the school year.  This has been challenging with the school is limited funding for supplies.
We are a classroom of 20 friendly and curious learners, from various ethnic backgrounds, facing challenges,
including poverty and developmental delays.    My students are future scholars, teachers, doctors, and acco
mplished human beings.  I need the public is help to raise money for materials that help maintain the atten
tion of my special needs students.Last year was my first year teaching Kindergarten inclusion.  I learned t
hat students can wiggle, and learn at the same time!    My students need sensory toys to maintain focus on
simple tasks that will shape their social and academic future.  My students with ADHD find themselves movin
g their hands, feet, and bodies without much control.  Sensory toys help my students use their energy in a
positive manner (fidget toys, bouncy chairs, etc).  With fidget toys, my students use their energy to play
appropriately while listening at the same time.    I have noticed that my students with special needs are a
ble to pay attention when they are given the proper tools and models to succeed.  My goal is to accommodate
young learners with special needs, and allow them to express themselves in a positive way that will lead to
their success now and in the future.
```

In [21]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
As an inclusion kindergarten teacher I am constantly looking for materials to help students develop and gro
w throughout the school year This has been challenging with the school is limited funding for supplies We a
re a classroom of 20 friendly and curious learners from various ethnic backgrounds facing challenges includ
ing poverty and developmental delays My students are future scholars teachers doctors and accomplished huma
n beings I need the public is help to raise money for materials that help maintain the attention of my spec
ial needs students Last year was my first year teaching Kindergarten inclusion I learned that students can
wiggle and learn at the same time My students need sensory toys to maintain focus on simple tasks that will
shape their social and academic future My students with ADHD find themselves moving their hands feet and bo
dies without much control Sensory toys help my students use their energy in a positive manner fidget toys b
ouncy chairs etc With fidget toys my students use their energy to play appropriately while listening at the
same time I have noticed that my students with special needs are able to pay attention when they are given
the proper tools and models to succeed My goal is to accommodate young learners with special needs and allo
w them to express themselves in a positive way that will lead to their success now and in the future
```

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', 'thos
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', '
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again'
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'fe
            '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', '
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mus
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren
            'won', "won't", 'wouldn', "wouldn't"]
```

In [23]:
```python
# 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 [00:31<00:00,
1583.45it/s]
```

In [24]:
```python
# after preprocesing
preprocessed_essays[200]
```

Out[24]:
```
'as inclusion kindergarten teacher i constantly looking materials help students develop grow throughout sch
ool year this challenging school limited funding supplies we classroom 20 friendly curious learners various
ethnic backgrounds facing challenges including poverty developmental delays my students future scholars tea
chers doctors accomplished human beings i need public help raise money materials help maintain attention sp
ecial needs students last year first year teaching kindergarten inclusion i learned students wiggle learn t
ime my students need sensory toys maintain focus simple tasks shape social academic future my students adhd
find moving hands feet bodies without much control sensory toys help students use energy positive manner fi
dget toys bouncy chairs etc with fidget toys students use energy play appropriately listening time i notice
d students special needs able pay attention given proper tools models succeed my goal accommodate young lea
rners special needs allow express positive way lead success future'
```

In [25]:
```python
project_data['preprocessed_essays'] = preprocessed_essays
```

1.4 Preprocessing of `project_title`

In [26]:
```python
# similarly you can preprocess the titles also
project_data.head(2)
```

Out[26]:

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

In [27]:

```python
# printing some random project titles.
print(project_data['project_title'].values[54])
print("="*50)
print(project_data['project_title'].values[89])
print("="*50)
print(project_data['project_title'].values[99])
print("="*50)
print(project_data['project_title'].values[156])
print("="*50)
print(project_data['project_title'].values[846])
print("="*50)
```

```
Swim For Life At YMCA!
==================================================
Education Through Technology
==================================================
Teaching Math With Manipulatives
==================================================
Getting Our MOVE On!
==================================================
21st Century Skills and Technology Optimized to Improve OUR World!!!
==================================================
```

In [28]:

```python
#Removing phrases from the title features
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"Gotta", "Got to", 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 [29]:

```python
#Checkingt titles after removing phrases
sent = decontracted(project_data['project_title'].values[836])
print(sent)
print("="*50)
```

```
Digital Magazine
==================================================
```

In [30]:

```python
# Remove \\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)
```

```
Digital Magazine
```

In [31]:

```python
#Removing numbers & symbols form the titles
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Digital Magazine
```

In [32]:
```
# 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', 'thos
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', '
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'f
            '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', '
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mus
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'were
            'won', "won't", 'wouldn', "wouldn't"]
```

In [33]:
```
#Combining all the above preprocessed statements
from tqdm import tqdm
preprocessed_titles = []
# 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 = 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%|████████████████████████████████████████████████████████████████████████| 50000/50000 [00:01<00:00, 2
9049.64it/s]
```

In [34]:
```
#checking cleaned text after preprocesing
print(preprocessed_titles[54])
print("="*50)
print(preprocessed_titles[89])
print("="*50)
print(preprocessed_titles[99])
print("="*50)
print(preprocessed_titles[156])
print("="*50)
print(preprocessed_titles[836])
```

```
swim for life at ymca
==================================================
education through technology
==================================================
teaching math with manipulatives
==================================================
getting our move on
==================================================
digital magazine
```

In [35]:
```
project_data['preprocessed_titles'] = preprocessed_titles
```

1.5 Preparing data for models

In [36]:
```
project_data.columns
```

Out[36]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', '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',
       'clean_categories', 'clean_subcategories', 'essay',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [37]:

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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sport
s', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (50000, 9)
```

In [38]:

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Gove
rnment', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingAr
ts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Heal
th_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_We
llness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (50000, 30)
```

In [39]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
vectorizer = CountVectorizer(binary=True)
school_state_count = vectorizer.fit_transform(project_data['school_state'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",school_state_count.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks', 'k
y', '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 matrix after one hot encodig  (50000, 51)
```

```
In [40]: #Replacing spaces & hyphens in the text of project grade category with underscore
         #converting Capital letters in the string to smaller letters
         #Performing avalue count of project grade category
         # https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-onp
         project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
         project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
         project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
         project_data['project_grade_category'].value_counts()
```

```
Out[40]: grades_prek_2    20316
         grades_3_5       16968
         grades_6_8        7750
         grades_9_12       4966
         Name: project_grade_category, dtype: int64
```

```
In [41]: #One hot encoding project grade category feature
         vectorizer = CountVectorizer(binary=True)
         project_grade_one = vectorizer.fit_transform(project_data['project_grade_category'].values)
         print(vectorizer.get_feature_names())
         print("Shape of matrix after one hot encoding ",project_grade_one.shape)
```

```
         ['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
         Shape of matrix after one hot encoding  (50000, 4)
```

```
In [42]: # check if we have any nan values are there in the column
         print(project_data['teacher_prefix'].isnull().values.any())
         print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
         True
         number of nan values 2
```

```
In [43]: #Replacing the Nan values with most frequent value in the column
         project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
```

```
In [44]: # check if we have any nan values are there in the column
         print(project_data['teacher_prefix'].isnull().values.any())
         print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
         False
         number of nan values 0
```

```
In [45]: #Converting teacher prefix text into smaller case
         project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()
         project_data['teacher_prefix'].value_counts()
```

```
Out[45]: mrs.       26142
         ms.        17936
         mr.         4859
         teacher     1061
         dr.            2
         Name: teacher_prefix, dtype: int64
```

```
In [46]: #One hot encoding the teacher prefix column
         vectorizer = CountVectorizer(binary=True)
         teacher_prefix_one = vectorizer.fit_transform(project_data['teacher_prefix'].values)
         print(vectorizer.get_feature_names())
         print("Shape of matrix after one hot encodig ",teacher_prefix_one.shape)
```

```
         ['dr', 'mr', 'mrs', 'ms', 'teacher']
         Shape of matrix after one hot encodig  (50000, 5)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [47]:
```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (50000, 12210)
```

In [48]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

In [49]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (50000, 12210)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [50]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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

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

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

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


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

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


'''
```

Out[50]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef (https://stackoverf
low.com/a/38230349/4084039\ndef) loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = ope
n(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.spl
it()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n
model[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel = loadGlo
veModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading Glove Model\n1917
495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ============================\n\nwords = []\nf
or 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 w
ords in the coupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number
 of words that are present in both glove vectors and our coupus",       len(inter_words),"(",np.round(len(i
nter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in word
s:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(words_cou
rpus))\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 (http://www.jessicayung.com/how-to-use-pickle-to-save-and-loa
d-variables-in-python/\n\nimport) pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words
_courpus, f)\n\n\n'
```

```
In [51]: # stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load
         # make sure you have the glove_vectors file
         with open('glove_vectors', 'rb') as f:
             model = pickle.load(f)
             glove_words =  set(model.keys())
```

```
In [52]: # average Word2Vec
         # compute average word2vec for each review.
         avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
         for sentence in tqdm(preprocessed_essays): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             cnt_words =0; # num of words with a valid vector in the sentence/review
             for word in sentence.split(): # for each word in a review/sentence
                 if word in glove_words:
                     vector += model[word]
                     cnt_words += 1
             if cnt_words != 0:
                 vector /= cnt_words
             avg_w2v_vectors.append(vector)

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

```
100%|███████████████████████████████████████████████████████████████████████████| 50000/50000 [00:22<00:00,
2262.17it/s]

50000
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

```
In [53]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
         tfidf_model = TfidfVectorizer()
         tfidf_model.fit(preprocessed_essays)
         # we are converting a dictionary with word as a key, and the idf as a value
         dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
         tfidf_words = set(tfidf_model.get_feature_names())
```

```
In [54]: # average Word2Vec
         # compute average word2vec for each review.
         tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
         for sentence in tqdm(preprocessed_essays): # for each review/sentence
             vector = np.zeros(300) # as word vectors are of zero length
             tf_idf_weight =0; # num of words with a valid vector in the sentence/review
             for word in sentence.split(): # for each word in a review/sentence
                 if (word in glove_words) and (word in tfidf_words):
                     vec = model[word] # getting the vector for each word
                     # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(
                     tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
                     vector += (vec * tf_idf) # calculating tfidf weighted w2v
                     tf_idf_weight += tf_idf
             if tf_idf_weight != 0:
                 vector /= tf_idf_weight
             tfidf_w2v_vectors.append(vector)

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

```
100%|███████████████████████████████████████████████████████████████████████████| 50000/50000 [02:21<00:00,
352.38it/s]

50000
300
```

In [55]:
```
# Similarly you can vectorize for title also
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_titles = set(tfidf_model.get_feature_names())
```

In [56]:
```
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf val
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

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

```
100%|█████████████████████████████████████████████████████████████████████████| 50000/50000 [00:02<00:00, 2
0685.23it/s]

50000
300
```

Concatenating Project Essay & Project Titles

In [57]:
```
project_data["concatenated_essays_titles"] = project_data["preprocessed_titles"].map(str) +\
                        project_data["preprocessed_essays"].map(str)
```

1.5.3 Vectorizing Numerical features

In [58]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [59]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardSc
from sklearn.preprocessing import StandardScaler

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

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

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

```
Mean : 299.33367619999996, Standard deviation : 378.20927190421384
```

In [60]:
```
price_standardized
```

Out[60]:
```
array([[-0.38268146],
       [-0.00088225],
       [ 0.57512161],
       ...,
       [-0.65382764],
       [-0.52109689],
       [ 0.54492668]])
```

Counting the number of words in Essays

In [61]:
```
project_data["preprocessed_essays"] = preprocessed_essays
```

In [62]:
```
essay_words_total = []
for _ in project_data["preprocessed_essays"]:
    x = len(_.split())
    essay_words_total.append(x)
```

In [63]:
```
project_data["essay_words_total"] = essay_words_total
```

Counting the number of words in Project Title

In [64]:
```
project_data["preprocessed_titles"] = preprocessed_titles
```

In [65]:
```
title_words_total = []
for _ in project_data["preprocessed_titles"]:
    y = len(_.split())
    title_words_total.append(y)
```

In [66]:
```
project_data["title_words_total"] = title_words_total
```

Calculate sentiment score for essays

In [67]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
neg = []
pos = []
neu = []
compound = []

for _ in tqdm(project_data["preprocessed_essays"]) :
    w = sid.polarity_scores(_)['neg']
    x = sid.polarity_scores(_)['pos']
    y = sid.polarity_scores(_)['neu']
    z = sid.polarity_scores(_)['compound']
    neg.append(w)
    pos.append(x)
    neu.append(y)
    compound.append(z)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000 [06:43<00:00,
123.81it/s]
```

In [68]:
```
project_data["pos"] = pos
```

In [69]:
```
project_data["neg"] = neg
```

In [70]:
```
project_data["neu"] = neu
```

In [71]:
```
project_data["compound"] = compound
```

In [72]:

```
project_data.head()
```

Out[72]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cat
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	IN	2016-12-05 13:43:57	grades_p
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr.	FL	2016-10-25 09:22:10	grades
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms.	AZ	2016-08-31 12:03:56	grades
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs.	KY	2016-10-06 21:16:17	grades_p
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	mrs.	TX	2016-07-11 01:10:09	grades_p

5 rows × 29 columns

In [ ]:

1.5.4 Merging all the above features

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

In [73]:

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

```
(50000, 9)
(50000, 30)
(50000, 12210)
(50000, 1)
```

In [74]:

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

Out[74]:

```
(50000, 12250)
```

In [75]:

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

__ Computing Sentiment Scores__

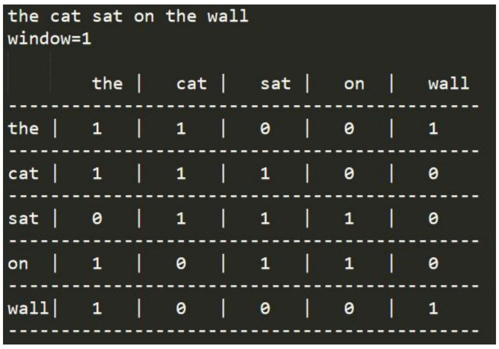
In [76]:

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

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

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students with the big
for learning my students learn in many different ways using all of our senses and multiple intelligences i us
of techniques to help all my students succeed students in my class come from a variety of different backgrou
for wonderful sharing of experiences and cultures including native americans our school is a caring community
learners which can be seen through collaborative student project based learning in and out of the classroom k
in my class love to work with hands on materials and have many different opportunities to practice a skill be
mastered having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten
montana is the perfect place to learn about agriculture and nutrition my students love to role play in our pr
in the early childhood classroom i have had several kids ask me can we try cooking with real food i will take
and create common core cooking lessons where we learn important math and writing concepts while cooking delic
food for snack time my students will have a grounded appreciation for the work that went into making the food
of where the ingredients came from as well as how it is healthy for their bodies this project would expand ou
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce make
and mix up healthy plants from our classroom garden in the spring we will also create our own cookbooks to be
shared with families students will gain math and literature skills as well as a life long enjoyment for healt
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,
```

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_` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref (https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/))



```
the cat sat on the wall
window=1

         the |  cat |  sat |  on  |  wall
---------------------------------------------
the |  1   |  1   |  0   |  0   |  1
---------------------------------------------
cat |  1   |  1   |  1   |  0   |  0
---------------------------------------------
sat |  0   |  1   |  1   |  1   |  0
---------------------------------------------
on  |  1   |  0   |  1   |  1   |  0
---------------------------------------------
wall|  1   |  0   |  0   |  0   |  1
---------------------------------------------
```

- step 3 Use TruncatedSVD (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html) on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/pca-code-example-using-non-visualization/)

> - 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 (https://www.kdnuggets.com/2017/03/simple-xgboost-tutorial-iris-dataset.html)**
- step 6:Hyper parameter tuning (Consider any two hyper parameters)
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) 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 [77]:

```
pip install XGBOOST
```

```
The following command must be run outside of the IPython shell:

    $ pip install XGBOOST

The Python package manager (pip) can only be used from outside of IPython.
Please reissue the `pip` command in a separate terminal or command prompt.

See the Python documentation for more information on how to install packages:

    https://docs.python.org/3/installing/ (https://docs.python.org/3/installing/)
```

In [78]:

```
import sys
import math

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

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

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, verbose_eval

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

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

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

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

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


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

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

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

Out[78]:

```
'clf = XGBoostClassifier(eval_metric = \'auc\', num_class = 2, nthread = 4,)\n############################
#######################################\n#               Change from here
#\n###################################################################\nparameters = {\n    \'num_boost_r
ound\': [100, 250, 500],\n    \'eta\': [0.05, 0.1, 0.3],\n    \'max_depth\': [6, 9, 12],\n    \'subsample
\': [0.9, 1.0],\n    \'colsample_bytree\': [0.9, 1.0],\n}\n\nclf = GridSearchCV(clf, parameters)\nX = np.
array([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])\nY = np.array([0, 1, 0, 1, 0, 1])\nclf.fit(X, Y)\n\n# p
rint(clf.grid_scores_)\nbest_parameters, score, _ = max(clf.grid_scores_, key=lambda x: x[1])\nprint(\'sc
ore:\', score)\nfor param_name in sorted(best_parameters.keys()):\n    print("%s: %r" % (param_name, best
_parameters[param_name]))'
```

2. TruncatedSVD

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

In [79]:

```
# 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
```

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

In [80]:

```
data = project_data
data.head(5)
```

Out[80]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cat
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	IN	2016-12-05 13:43:57	grades_p
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr.	FL	2016-10-25 09:22:10	grades
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms.	AZ	2016-08-31 12:03:56	grades
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs.	KY	2016-10-06 21:16:17	grades_p
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	mrs.	TX	2016-07-11 01:10:09	grades_p

5 rows × 29 columns

In [81]:
```
y = data['project_is_approved'].values
data.drop(['project_is_approved'], axis=1, inplace=True)
data.head(1)
```

Out[81]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_categ
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	IN	2016-12-05 13:43:57	grades_pre

1 rows × 28 columns

In [82]:
```
X = data
```

In [83]:
```
# check if we have any nan values are there in the column
print(X['teacher_prefix'].isnull().values.any())
print("number of nan values",X['teacher_prefix'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [84]:
```
#Converting teacher prefix text into smaller case
X['teacher_prefix'] = X['teacher_prefix'].str.lower()
X['teacher_prefix'].value_counts()
```

Out[84]:
```
mrs.       26142
ms.        17936
mr.         4859
teacher     1061
dr.            2
Name: teacher_prefix, dtype: int64
```

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

In [85]:
```
#Splitting data into test & train set
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
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)
```

In [86]:
```
#Splitting training data into training & cross validation sets
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,
stratify= y_train,
test_size = 0.33)
```

In [87]:
```python
# printing some random reviews
print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[100])
print("="*50)
print(X_train['essay'].values[300])
print("="*50)
print(X_train['essay'].values[200])
print("="*50)
print(X_train['essay'].values[500])
print("="*50)
```

```
I have an incredible classroom filled with the beautiful hearts and minds of eager learners. Each day we tr
y our best to make a day of learning and fun through engaging lessons and activities, and peer-to-peer lear
ning. We are becoming mathematicians, readers, scientists, writers, and citizens of our community and the w
orld, and we do our best each day to spread love and kindness to all. \r\n\r\nStudents come from different 
places before school and leave to many places after school. No matter where they come from, or where they g
o, when they are with me, learning is our number one goal. Though my time with them is only bell-to-bell, I
hope that my love and dedication to them extends past the school day, and they know how much they are loved
and appreciated every time they think of me.Our classroom has been blessed with many gift this year - numer
ous volunteers, 4th grade buddies, and tons of opportunities to connect with the materials that have alread
y been donated to us from previous donors. About this time of year, the big push to the finish comes, and I
like to evaluate what major needs my students still have. We also like to brainstorm what we'd like to see 
more of, or have more of in our classroom. \r\nI realized that my students still need work increasing their
fine motor skills, and they desire a comfy place to read within our classroom library. Therefore, I am tryi
ng to get both accomplished in this project. I am requesting some more hands on activities for increasing f
ine motor skills, including pinching and grasping (also throwing in some math!). My students also voiced th
at they would love a place to snuggle up and read - while our classroom library has TONS of books (we are s
o lucky!) we don't have a place to really settle in with a good book. We are hoping this project fills so 
we can get what we need, and what we want!
==================================================
I teach fifth grade at suburban school in Oklahoma. My students come from various cultural and socioeconomi
c backgrounds. They are a diverse and unique bunch. Each day we strive to do the very best we can. We are a
Title I school and a large percentage of our students qualify for free and reduced-price lunch. \r\n\r\nOur
school is a learning community. We try to meet the needs of every students and challenge them to set and re
ach goals each and every day.Small group instruction is vital to any literacy instruction. The chance to wo
rk one-on-one or within a small group is invaluable to increasing literacy. These materials will enhance my
small group literacy instruction while incorporating various science and social studies standards. These te
xts are not only educational from a science and social studies perspective, but also are high-interest text
s. For example, when students are reading a comic book about the American Revolution or the scientific meth
od, they are much more interested in engaged. When engagement happens, learning happens.  These high-intere
st texts will allow students, not only to sharpen their literacy and fluency skills, but also thoroughly en
gage them in our science and social studies objectives.
==================================================
The ESOL students that I serve are very eager to learn and explore new concepts. When they learn new standa
rds they are very excited. They enjoy collaborating with each other on assignments and projects. In additio
n to collaborating with each other, they especially enjoy doing hands-on projects that require them to use 
technology.\r\nMy students come from a high poverty area where technology is very limited. The students liv
e in the suburbs of a metro area where 100% of them receive free lunch and breakfast. Our school strives to
provide our students with multiple opportunities for success and growth. This is why the need for technolog
y is essential for my students to flourish.I am asking for three Samsung Tablets so that I can expose and e
ngage my ESL students with more hands-on activities that involves technology. For instance, in my classroom
the tablets will allow students an opportunity to research and create presentations through online programs
like Powtoons or Puppet Pals in a creative way. I also like for the students to use QR codes in many of our
small group lessons and the tablets will be essential to those lessons. This will also help students by  re
inforcing the English Language.\r\nExposing students to more hands-on activities that involves technology c
an help my ESL students stay more engaged in their learning. Also, it will allow them more opportunities fo
r my students to use different programs that will help build and enrich their vocabulary and writing skills
in the English Language.\r\n
==================================================
I work in a Title I school in North Carolina. My students are typically from low income families. Most come
to school each day with a smile ready to learn! They want to learn by doing and sharing activities with the
ir peers. We are a school that practices inquiry based learning which requires many materials for them to e
xplore. \r\nFirst grade is an important year in creating a love of learning that will last a lifetime. This
is why it is important to me to create exciting and engaging lesson plans. I strive to help the students wo
nder and ask questions ad make discoveries each day. Please consider helping me accomplish this by donating
to my classroom.Studies show that keeping your blood flowing helps aid concentration and retention of infor
mation. However, our education system is structured in a way that creates a lot of time for sitting at desk
s working quietly. This is a difficult conundrum for today's teacher to try to overcome.\r\nThese bands wil
l help the students be able to get some energy out and keep their blood flowing while sitting at their desk
s.  This will allow them to comply with the task of sitting and staying focused while exercising their musc
les under their desks. This is especially important for the young learners in my classroom.
==================================================
These students are full of ideas and imaginative feats they cannot complete alone. I want the classroom to 
capture their attention and help them be endlessly creative.  The school has a Title 1 code that provides a
lmost 80% free or reduced lunch. \r\n\r\nMost of the students have never traveled outside of this small tow
```

```
n and will most likely continue this generational trend. The involvement of parents grows increasingly diff
icult because of this day and age.  Students here face challenges unlike the average child and any comfort
not afforded at home, I hope they discover at the classroom.\r\nMy students are at a crucial age where a sc
hool can become a place of boredom or a springboard for future success.The materials requested will update
our Makerspace Library.  New bits will allow them to explore new facets of engineering and design.  The kit
s provided will give the students the ability to engineer, create, and discover inventions they have only d
reamed about. It will also enable the students to discover the \"Why\"s and \"How\"s mechanical devices fun
ction. Instead of replying with \"It just does\" or \"I don't know\" they will reply with what they have ph
ysically discovered themselves!\r\n\r\n Students love the STEM activities I am able to do with Littlebits a
nd several students have even developed a love for engineering, math, and science.  Students with no intere
st in coming to school, ask if they can come early or stay late to investigate new inventions or perfect ol
der ones!  It's seriously, the coolest thing I've ever seen.
==================================================
```

In [88]:

```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 [89]:

```python
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative, and hard
-working.  They are all unique - unique in their interests, their learning, their abilities, and so much mo
re.  What they all have in common is their desire to learn each day, despite difficulties that they encount
er.  \r\nOur classroom is amazing - because we understand that everyone learns at their own pace.  As the t
eacher, I pride myself in making sure my students are always engaged, motivated, and inspired to create the
ir own learning! \r\nThis project is to help my students choose seating that is more appropriate for them,
developmentally.  Many students tire of sitting in chairs during lessons, and having different seats availa
ble helps to keep them engaged and learning.\r\nFlexible seating is important in our classroom, as many of
our students struggle with attention, focus, and engagement.  We currently have stability balls for seatin
g, as well as regular chairs, but these stools will help students who have trouble with balance, or find it
difficult to sit on a stability ball for a long period of time.  We are excited to try these stools as a pa
rt of our engaging classroom community!
==================================================
```

In [90]:

```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)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative, and hard
-working.  They are all unique - unique in their interests, their learning, their abilities, and so much mo
re.  What they all have in common is their desire to learn each day, despite difficulties that they encount
er.    Our classroom is amazing - because we understand that everyone learns at their own pace.  As the tea
cher, I pride myself in making sure my students are always engaged, motivated, and inspired to create their
own learning!   This project is to help my students choose seating that is more appropriate for them, devel
opmentally.  Many students tire of sitting in chairs during lessons, and having different seats available h
elps to keep them engaged and learning.  Flexible seating is important in our classroom, as many of our stu
dents struggle with attention, focus, and engagement.  We currently have stability balls for seating, as we
ll as regular chairs, but these stools will help students who have trouble with balance, or find it difficu
lt to sit on a stability ball for a long period of time.  We are excited to try these stools as a part of o
ur engaging classroom community!
```

```
In [91]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
         sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
         print(sent)
```

```
Describing my students is not an easy task Many would say that they are inspirational creative and hard wor
king They are all unique unique in their interests their learning their abilities and so much more What the
y all have in common is their desire to learn each day despite difficulties that they encounter Our classro
om is amazing because we understand that everyone learns at their own pace As the teacher I pride myself in
making sure my students are always engaged motivated and inspired to create their own learning This project
is to help my students choose seating that is more appropriate for them developmentally Many students tire
of sitting in chairs during lessons and having different seats available helps to keep them engaged and lea
rning Flexible seating is important in our classroom as many of our students struggle with attention focus
and engagement We currently have stability balls for seating as well as regular chairs but these stools wil
l help students who have trouble with balance or find it difficult to sit on a stability ball for a long pe
riod of time We are excited to try these stools as a part of our engaging classroom community
```

```
In [92]: # 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', 'thos
                     'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'c
                     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'c
                     'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before',
                     'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again',
                     'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'fe
                     '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', 'c
                     've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
                     "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'must
                     "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren
                     'won', "won't", 'wouldn', "wouldn't"]
```

Preprocessing for Train Data

```
In [93]: # Combining all the above stundents
         from tqdm import tqdm
         preprocessed_essays_xtr = []
         # tqdm is for printing the status bar
         for sentance in tqdm(X_train['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.lower() not in stopwords)
             preprocessed_essays_xtr.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 22445/22445 [00:14<00:00,
1570.99it/s]
```

```
In [94]: # after preprocesing
         preprocessed_essays_xtr[300]
```

```
Out[94]: 'esol students serve eager learn explore new concepts learn new standards excited enjoy collaborating assig
         nments projects addition collaborating especially enjoy hands projects require use technology students come
         high poverty area technology limited students live suburbs metro area 100 receive free lunch breakfast scho
         ol strives provide students multiple opportunities success growth need technology essential students flouri
         sh asking three samsung tablets expose engage esl students hands activities involves technology instance cl
         assroom tablets allow students opportunity research create presentations online programs like powtoons pupp
         et pals creative way also like students use qr codes many small group lessons tablets essential lessons als
         o help students reinforcing english language exposing students hands activities involves technology help es
         l students stay engaged learning also allow opportunities students use different programs help build enrich
         vocabulary writing skills english language'
```

In [95]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_xcv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['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.lower() not in stopwords)
    preprocessed_essays_xcv.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████| 11055/11055 [00:07<00:00,
1555.77it/s]
```

In [96]:
```
# after preprocesing
preprocessed_essays_xcv[300]
```

Out[96]:
```
'eclectic group students wide range interests skills smaller campus still growing phase students excepting
new students come campus students add anytime throughout first three quarters school year school hybrid sch
ool class fully online learning experience gives students unique opportunity experience educational technol
ogy since consistently using technology school part high poverty community basic technology still required
helping project helping provide math basic supplies students need math geometry students currently share tw
o compasses protractor teaching drawing circles angles hard students not understand concepts without suppli
es hard see certain techniques put practice supplies allow students see math concepts techniques come life
constructing specific types shapes angels vital comprehension material list items consist supplies neither
students bring classroom would greatly affect learning positive way'
```

In [97]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_xte = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['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.lower() not in stopwords)
    preprocessed_essays_xte.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████| 16500/16500 [00:10<00:00,
1568.76it/s]
```

In [98]:
```
# after preprocesing
preprocessed_essays_xte[300]
```

Out[98]:
```
'jms middle school approximately 440 6th 7th 8th graders 60 students free reduced lunches estimate another
10 similar need 30 hispanic 20 african american 50 caucasian diverse community partners mentor many student
s help meet social emotional needs connecting positive male adult life needs area continue grow annually bo
ard games generously donate towards enable many teachers connect students lunch social emotional basis conn
ections pay classroom better relationships teachers students fewer disciplinary problems ultimately engaged
learning throughout week generation often described connected digital sense yet lacks healthy intimacy good
friendships offer needed art conversation shared experiences oftentimes nonexistent face time teachers stud
ents gives kids chance feel valued cared invested students care much know know much care board game lunch g
roup offers opportunity students experience needs much many students interactions board game lunch group ma
y time truly connect adult week please consider donation much needed area students lives teachers donate ti
me attention diverse economically impoverished 8th grade student population look forward sharing unique out
comes experiences connections laughter group takes shape'
```

In [99]:

```
# similarly you can preprocess the titles also
#printing random titles
print(data['project_title'].values[49])
print("="*50)
print(data['project_title'].values[89])
print("="*50)
print(data['project_title'].values[999])
print("="*50)
print(data['project_title'].values[1116])
print("="*50)
print(data['project_title'].values[2000])
print("="*50)
```

```
Rainy Day Run Around!
==================================================
Education Through Technology
==================================================
Focus Pocus
==================================================
Safety in the Science Lab!
==================================================
Steady Stools for Active Learning
==================================================
```

In [100]:

```
#Removing phrases from the title features
import re
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"Gotta", "Got to", 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 [101]:

```
#Checkingt titles after removing phrases
sent = decontracted(project_data['project_title'].values[896])
print(sent)
print("="*50)
```

```
A kidney table for small group instruction
==================================================
```

In [102]:

```
# Remove \\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)
```

```
A kidney table for small group instruction
```

In [103]:

```
# 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', 'thos
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', '
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again'
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'f
            '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', '
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mus
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren
            'won', "won't", 'wouldn', "wouldn't"]
```

In [104]:

```
preprocessed_titles_xtr = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['project_title'].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_titles_xtr.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445 [00:00<00:00, 2
7256.42it/s]
```

In [105]:

```
#checking cleaned text after preprocesing
print(preprocessed_titles_xtr[89])
print("="*50)
```

```
creativity problem solving oh my
==================================================
```

In [106]:

```
preprocessed_titles_xcv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['project_title'].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_titles_xcv.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11055/11055 [00:00<00:00, 2
4565.70it/s]
```

In [107]:

```
print(preprocessed_titles_xcv[89])
print("="*50)
```

```
come read all about it time for kid magazines needed
==================================================
```

In [108]:
```
preprocessed_titles_xte = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['project_title'].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_titles_xte.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 16500/16500 [00:00<00:00, 2
8446.88it/s]
```

In [109]:
```
print(preprocessed_titles_xte[89])
print("="*50)
```

```
visual models handwriting
==================================================
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [110]:
```
#We use fit only for train data
vectorizer_state = CountVectorizer(binary=True)
vectorizer_state.fit(X_train['school_state'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer_state.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer_state.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer_state.transform(X_test['school_state'].values)
print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer_state.get_feature_names())
print("="*75)
```

```
After vectorizations
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks', 'k
y', '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']
===========================================================================
```

2.2.2 One hot encoding the categorical features : teacher_prefix

In [111]:
```
# we use count vectorizer to convert the values into one
#We use fit only for train data
vectorizer_tp = CountVectorizer(binary=True)
vectorizer_tp.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer_tp.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer_tp.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer_tp.transform(X_test['teacher_prefix'].values)
print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer_tp.get_feature_names())
print("="*50)
```

```
After vectorizations
(22445, 5) (22445,)
(11055, 5) (11055,)
(16500, 5) (16500,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
==================================================
```

2.2.3 One hot encoding the categorical features : grades

In [112]:

```
#Replacing spaces & hyphens in the text of project grade category with underscore
#converting Capital letters in the string to smaller letters
#Performing avalue count of project grade category
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-onproj
data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[112]:

```
grades_prek_2    20316
grades_3_5       16968
grades_6_8        7750
grades_9_12       4966
Name: project_grade_category, dtype: int64
```

In [113]:

```
#We use fit only for train data
vectorizer_grade = CountVectorizer()
vectorizer_grade.fit(X_train['project_grade_category'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer_grade.transform(X_train['project_grade_category'].values)
X_cv_grade_ohe = vectorizer_grade.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe = vectorizer_grade.transform(X_test['project_grade_category'].values)
print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer_grade.get_feature_names())
print("="*70)
```

```
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
======================================================================
```

2.2.4 One hot encoding the categorical features : project subject category

In [114]:

```
#We use fit only for train data
vectorizer_category = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), binary=True)
vectorizer_category.fit(X_train['clean_categories'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_cat_ohe = vectorizer_category.transform(X_train['clean_categories'].values)
X_cv_cat_ohe = vectorizer_category.transform(X_cv['clean_categories'].values)
X_test_cat_ohe = vectorizer_category.transform(X_test['clean_categories'].values)
print("After vectorizations")
print(X_train_cat_ohe.shape, y_train.shape)
print(X_cv_cat_ohe.shape, y_cv.shape)
print(X_test_cat_ohe.shape, y_test.shape)
print(vectorizer_category.get_feature_names())
print("="*70)
```

```
After vectorizations
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sport
s', 'Math_Science', 'Literacy_Language']
======================================================================
```

In [115]:

```
#We use fit only for train data
vectorizer_subcat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), binary=True)
vectorizer_subcat.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_subcat_ohe = vectorizer_subcat.transform(X_train['clean_subcategories'].values)
X_cv_subcat_ohe = vectorizer_subcat.transform(X_cv['clean_subcategories'].values)
X_test_subcat_ohe = vectorizer_subcat.transform(X_test['clean_subcategories'].values)
print("After vectorizations")
print(X_train_subcat_ohe.shape, y_train.shape)
print(X_cv_subcat_ohe.shape, y_cv.shape)
print(X_test_subcat_ohe.shape, y_test.shape)
print(vectorizer_subcat.get_feature_names())
print("="*70)
```

```
After vectorizations
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Gove
rnment', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingAr
ts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Heal
th_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_We
llness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
======================================================================
```

1.5.3 Vectorizing Numerical features

For Price feature

In [116]:

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

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

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

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

In [117]:

```
X_train_price_norm = X_train_price_norm.T
X_cv_price_norm = X_cv_price_norm.T
X_test_price_norm = X_test_price_norm.T

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

```
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

For Quantity

In [118]:

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

X_train_quantity_norm = quan_normalizer.transform(X_train['quantity'].values.reshape(1,-1))
X_cv_quantity_norm = quan_normalizer.transform(X_cv['quantity'].values.reshape(1,-1))
X_test_quantity_norm = quan_normalizer.transform(X_test['quantity'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

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

In [119]:

```
X_train_quantity_norm = X_train_quantity_norm.T
X_cv_quantity_norm = X_cv_quantity_norm.T
X_test_quantity_norm = X_test_quantity_norm.T

print("Final Matrix")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

For teacher previously posted projects

In [120]:

```
# Normalizing teacher previously posted projects
from sklearn.preprocessing import Normalizer
tpp_normalizer = Normalizer()
# normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].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.
tpp_normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

X_train_tpp_norm = tpp_normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.r
X_cv_tpp_norm = tpp_normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape
X_test_tpp_norm = tpp_normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.resh

print("After vectorizations")
print(X_train_tpp_norm.shape, y_train.shape)
print(X_cv_tpp_norm.shape, y_cv.shape)
print(X_test_tpp_norm.shape, y_test.shape)
print("="*100)
```

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

In [121]:

```
X_train_tpp_norm = X_train_tpp_norm.T
X_cv_tpp_norm = X_cv_tpp_norm.T
X_test_tpp_norm = X_test_tpp_norm.T

print(X_train_tpp_norm.shape, y_train.shape)
print(X_cv_tpp_norm.shape, y_cv.shape)
print(X_test_tpp_norm.shape, y_test.shape)
print("="*100)
```

```
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

2.2 Computing Co-occurance matrix

In [122]:

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

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

In [123]:

```
total_txt=[]
for i in range(len(X_train)):
    total_txt.append(preprocessed_essays_xtr[i]+preprocessed_titles_xtr[i])
```

In [124]:

```
len(total_txt)
```

Out[124]:

```
22445
```

Performing TFIDF Vectorization on Essays / Title Features

In [125]:
```
vectorizer = TfidfVectorizer(min_df=10,use_idf=True,stop_words=stopwords)
vectorizer.fit(total_txt)
vectorizer.transform(total_txt)
```

Out[125]:
```
<22445x8906 sparse matrix of type '<class 'numpy.float64'>'
        with 2222112 stored elements in Compressed Sparse Row format>
```

In [126]:
```
ft_names = vectorizer.get_feature_names()
scores = vectorizer.idf_
```

In [127]:
```
ft_names[300:310]
```

Out[127]:
```
['administrator',
 'administrators',
 'admirable',
 'admire',
 'admit',
 'admitted',
 'adobe',
 'adolescence',
 'adolescent',
 'adolescents']
```

In [128]:
```
top_2k = pd.DataFrame({"fts":ft_names,"idf_values":vectorizer.idf_})
```

In [129]:
```
top_2k=top_2k.sort_values(by=['idf_values'],ascending=False)
```

In [130]:
```
type(top_2k)
```

Out[130]:
```
pandas.core.frame.DataFrame
```

In [131]:
```
top_2k[200:210]
```

Out[131]:

	fts	idf_values
4145	indicator	8.620972
6935	saddens	8.620972
3793	hd	8.620972
5036	memoir	8.620972
6373	quenched	8.620972
6307	pt	8.620972
5792	pedagogy	8.620972
5826	perfection	8.620972
2563	duel	8.620972
5825	perfecting	8.620972

In [132]:
```
idf_scores = []
for value in range(len(scores)):
    idf_scores.append([scores[value],ft_names[value]])
```

In [133]:
```
idf_scores.sort(reverse = True)
idf_scores = idf_scores[0:2000]
```

In [134]:
```
idf_scores[0:10]
```

Out[134]:
```
[[8.62097243077783, 'zen'],
 [8.62097243077783, 'yearbooks'],
 [8.62097243077783, 'xtramath'],
 [8.62097243077783, 'woven'],
 [8.62097243077783, 'winters'],
 [8.62097243077783, 'winn'],
 [8.62097243077783, 'williams'],
 [8.62097243077783, 'whys'],
 [8.62097243077783, 'walker'],
 [8.62097243077783, 'wagon']]
```

In [135]:
```
top_2k_fts = []
for i in range(2000):
    top_2k_fts.append(idf_scores[i][1])
```

In [136]:
```
top_2k_fts
```

Out[136]:
```
['zen',
 'yearbooks',
 'xtramath',
 'woven',
 'winters',
 'winn',
 'williams',
 'whys',
 'walker',
 'wagon',
 'visitor',
 'versa',
 'verify',
 'varieties',
 'validated',
 'vacations',
 'va',
 'useless',
 'upwards',
 'upfront',
```

In [137]:
```
type(top_2k_fts)
```

Out[137]:
```
list
```

In [138]:
```
#code source - https://stackoverflow.com/questions/41661801/python-calculate-the-co-occurrence-matrix
L=len(top_2k_fts)
co_mat=np.zeros([L,L])
window=5
for sent in tqdm(total_txt):
    wrds=sent.split()
    for i,word in enumerate(wrds):
        if word in top_2k_fts:
            for j in range(max(i-window,0),min(i+window,len(wrds)-1)+1):
                if wrds[j] in top_2k_fts:
                    co_mat[top_2k_fts.index(word)][top_2k_fts.index(wrds[j])]+=1
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445 [01:30<00:00,
249.04it/s]
```

In [139]:
```
co_mat.shape
```

Out[139]:
```
(2000, 2000)
```

In [140]:
```
#https://docs.scipy.org/doc/numpy/reference/generated/numpy.fill_diagonal.html
np.fill_diagonal(co_mat, 0)
```

```
In [141]: co_mat
```

```
Out[141]: array([[0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 ...,
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.],
                 [0., 0., 0., ..., 0., 0., 0.]])
```

Testing the above code for co-occurence matrix for with a sample corpus & vocab

```
In [142]: corpus = ["abc def ijk pqr", "pqr klm opq","lmn pqr xyz abc def pqr abc"]
          top_words = ["abc", "pqr", "def"]
          wdw = 2
```

```
In [143]: #code source - https://stackoverflow.com/questions/41661801/python-calculate-the-co-occurrence-matrix
          T=len(top_words)
          Test=np.zeros([T,T])
          wdw=2
          for sent in tqdm(corpus):
              wrds=sent.split()
              for i,word in enumerate(wrds):
                  if word in top_words:
                      for j in range(max(i-wdw,0),min(i+wdw,len(wrds)-1)+1):
                          if wrds[j] in top_words:
                              Test[top_words.index(word)][top_words.index(wrds[j])]+=1
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 3/3 [00:
00<?, ?it/s]
```

```
In [144]: Test
```

```
Out[144]: array([[3., 3., 3.],
                 [3., 4., 2.],
                 [3., 2., 2.]])
```

```
In [145]: #https://docs.scipy.org/doc/numpy/reference/generated/numpy.fill_diagonal.html
          np.fill_diagonal(Test, 0)
```

```
In [146]: Test
```

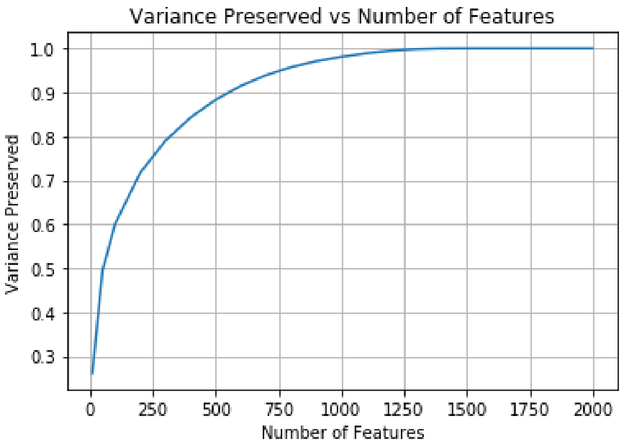
```
Out[146]: array([[0., 3., 3.],
                 [3., 0., 2.],
                 [3., 2., 0.]])
```

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

```
In [147]: # please write all the code with proper documentation, and proper titles for each subsection
          # go through documentations and blogs before you start coding
          # first figure out what to do, and then think about how to do.
          # reading and understanding error messages will be very much helpfull in debugging your code
          # make sure you featurize train and test data separatly

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

In [148]:
```
#https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
#Initially we considered 5000 features but due to memory issue we are limiting the number of features to 1
from sklearn.decomposition import TruncatedSVD
Number_of_features = [10,50,100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,1400,1500,1600,1700,1800
variance_preserved = []

for i in tqdm(Number_of_features):
    svd = TruncatedSVD(n_components=i, n_iter=2)
    svd.fit(co_mat)
    variance_preserved.append(svd.explained_variance_ratio_.sum())

plt.plot(Number_of_features,variance_preserved)
plt.xlabel("Number of Features")
plt.ylabel("Variance Preserved")
plt.title("Variance Preserved vs Number of Features")
plt.grid()
plt.show()
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 24/24 [01:44<00:0
0,  8.72s/it]
```



From the above plot we can observe that approximately 95% of variance is preserved with 1750 Features.

In [149]:
```
from sklearn.decomposition import TruncatedSVD
trun_svd = TruncatedSVD(n_components=1750,random_state=34)
trun_cooc_matrix = trun_svd.fit_transform(co_mat)
trun_cooc_matrix.shape
```

Out[149]:
```
(2000, 1750)
```

In [150]:
```
glove = {}
for words in range(len(top_2k_fts)):
    glove[top_2k_fts[i]] = trun_cooc_matrix[i]
```

In [151]:
```
# making a set of glove words for avg w2v
glove_words = set(glove.keys())
```

Vectorizing Essay text for Training set

In [152]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essay_xtr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(1750) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += glove[word]
        cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essay_xtr.append(vector)

print(len(avg_w2v_essay_xtr))
print(len(avg_w2v_essay_xtr[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 22445/22445 [00:01<00:00, 1
4106.42it/s]

22445
1750
```

Vectorizing Essay text for Test set

In [153]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essay_xte = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(1750) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += glove[word]
        cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essay_xte.append(vector)

print(len(avg_w2v_essay_xte))
print(len(avg_w2v_essay_xte[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 16500/16500 [00:01<00:00, 1
3714.42it/s]

16500
1750
```

Vectorizing Project Title for Train set

In [154]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_title_xtr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(1750) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += glove[word]
        cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_title_xtr.append(vector)

print(len(avg_w2v_title_xtr))
print(len(avg_w2v_title_xtr[0]))
```

```
100%|█████████████████████████████████████████████████████████████████████████| 22445/22445 [00:00<00:00, 4
3579.29it/s]

22445
1750
```

Vectorizing Project Title for Test set

In [155]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_title_xte = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(1750) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += glove[word]
        cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_title_xte.append(vector)

print(len(avg_w2v_title_xte))
print(len(avg_w2v_title_xte[0]))
```

```
100%|█████████████████████████████████████████████████████████████████████████| 16500/16500 [00:00<00:00, 4
4351.57it/s]

16500
1750
```

Normalising Essay word count

In [156]:

```
from sklearn.preprocessing import Normalizer
essay_words_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
essay_words_norm.fit(X_train['essay_words_total'].values.reshape(1,-1))

X_train_ewords_norm = essay_words_norm.transform(X_train['essay_words_total'].values.reshape(1,-1))
X_cv_ewords_norm = essay_words_norm.transform(X_cv['essay_words_total'].values.reshape(1,-1))
X_test_ewords_norm = essay_words_norm.transform(X_test['essay_words_total'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_ewords_norm.shape, y_train.shape)
print(X_cv_ewords_norm.shape, y_cv.shape)
print(X_test_ewords_norm.shape, y_test.shape)
print("="*100)
```

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

In [157]:

```
X_train_ewords_norm = X_train_ewords_norm.T
X_cv_ewords_norm = X_cv_ewords_norm.T
X_test_ewords_norm = X_test_ewords_norm.T

print("Final Matrix")
print(X_train_ewords_norm.shape, y_train.shape)
print(X_cv_ewords_norm.shape, y_cv.shape)
print(X_test_ewords_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalising Project Title word count

In [158]:

```
from sklearn.preprocessing import Normalizer
title_words_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
title_words_norm.fit(X_train['title_words_total'].values.reshape(1,-1))

X_train_twords_norm = title_words_norm.transform(X_train['title_words_total'].values.reshape(1,-1))
X_cv_twords_norm = title_words_norm.transform(X_cv['title_words_total'].values.reshape(1,-1))
X_test_twords_norm = title_words_norm.transform(X_test['title_words_total'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_twords_norm.shape, y_train.shape)
print(X_cv_twords_norm.shape, y_cv.shape)
print(X_test_twords_norm.shape, y_test.shape)
print("="*100)
```

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

In [159]:

```
X_train_twords_norm = X_train_twords_norm.T
X_cv_twords_norm = X_cv_twords_norm.T
X_test_twords_norm = X_test_twords_norm.T

print("Final Matrix")
print(X_train_twords_norm.shape, y_train.shape)
print(X_cv_twords_norm.shape, y_cv.shape)
print(X_test_twords_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalising Essay Sentiment scores

Normalising positive score

In [160]:

```
from sklearn.preprocessing import Normalizer
senti_pos_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
senti_pos_norm.fit(X_train['pos'].values.reshape(1,-1))

X_train_pos_norm = senti_pos_norm.transform(X_train['pos'].values.reshape(1,-1))
X_cv_pos_norm = senti_pos_norm.transform(X_cv['pos'].values.reshape(1,-1))
X_test_pos_norm = senti_pos_norm.transform(X_test['pos'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_pos_norm.shape, y_train.shape)
print(X_cv_pos_norm.shape, y_cv.shape)
print(X_test_pos_norm.shape, y_test.shape)
print("="*100)
```

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

In [161]:

```
X_train_pos_norm = X_train_pos_norm.T
X_cv_pos_norm = X_cv_pos_norm.T
X_test_pos_norm = X_test_pos_norm.T

print("Final Matrix")
print(X_train_pos_norm.shape, y_train.shape)
print(X_cv_pos_norm.shape, y_cv.shape)
print(X_test_pos_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalising Negative score

In [162]:

```python
from sklearn.preprocessing import Normalizer
senti_neg_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
senti_neg_norm.fit(X_train['neg'].values.reshape(1,-1))

X_train_neg_norm = senti_neg_norm.transform(X_train['neg'].values.reshape(1,-1))
X_cv_neg_norm = senti_neg_norm.transform(X_cv['neg'].values.reshape(1,-1))
X_test_neg_norm = senti_neg_norm.transform(X_test['neg'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_neg_norm.shape, y_train.shape)
print(X_cv_neg_norm.shape, y_cv.shape)
print(X_test_neg_norm.shape, y_test.shape)
print("="*100)
```

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

In [163]:

```python
X_train_neg_norm = X_train_neg_norm.T
X_cv_neg_norm = X_cv_neg_norm.T
X_test_neg_norm = X_test_neg_norm.T

print("Final Matrix")
print(X_train_neg_norm.shape, y_train.shape)
print(X_cv_neg_norm.shape, y_cv.shape)
print(X_test_neg_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalising Neutral scores

In [164]:

```python
from sklearn.preprocessing import Normalizer
senti_neu_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
senti_neu_norm.fit(X_train['neu'].values.reshape(1,-1))

X_train_neu_norm = senti_neu_norm.transform(X_train['neu'].values.reshape(1,-1))
X_cv_neu_norm = senti_neu_norm.transform(X_cv['neu'].values.reshape(1,-1))
X_test_neu_norm = senti_neu_norm.transform(X_test['neu'].values.reshape(1,-1))
print("After vectorizations")

print(X_train_neu_norm.shape, y_train.shape)
print(X_cv_neu_norm.shape, y_cv.shape)
print(X_test_neu_norm.shape, y_test.shape)
print("="*100)
```

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

In [165]:

```
X_train_neu_norm = X_train_neu_norm.T
X_cv_neu_norm = X_cv_neu_norm.T
X_test_neu_norm = X_test_neu_norm.T

print("Final Matrix")
print(X_train_neu_norm.shape, y_train.shape)
print(X_cv_neu_norm.shape, y_cv.shape)
print(X_test_neu_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Normalising Compound scores

In [166]:

```
from sklearn.preprocessing import Normalizer
senti_comp_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.

senti_comp_norm.fit(X_train['compound'].values.reshape(1,-1))
X_train_comp_norm = senti_comp_norm.transform(X_train['compound'].values.reshape(1,-1))
X_cv_comp_norm = senti_comp_norm.transform(X_cv['compound'].values.reshape(1,-1))
X_test_comp_norm = senti_comp_norm.transform(X_test['compound'].values.reshape(1,-1))
print("After vectorizations")
print(X_train_comp_norm.shape, y_train.shape)
print(X_cv_comp_norm.shape, y_cv.shape)
print(X_test_comp_norm.shape, y_test.shape)
print("="*100)
```

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

In [167]:

```
X_train_comp_norm = X_train_comp_norm.T
X_cv_comp_norm = X_cv_comp_norm.T
X_test_comp_norm = X_test_comp_norm.T

print("Final Matrix")
print(X_train_comp_norm.shape, y_train.shape)
print(X_cv_comp_norm.shape, y_cv.shape)
print(X_test_comp_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

2.4 Merge the features from step 3 and step 4

In [168]:

```
# 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
```

Merging Numerical & Categorical features

In [169]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_norm, X_train_ca
X_te_set = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_price_norm, X_test_cat_ohe

print("Final Data matrix")
print(X_tr_set.shape, y_train.shape)
print(X_te_set.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 3608) (22445,)
(16500, 3608) (16500,)
====================================================================================================
```

2.5 Apply XGBoost on the Final Features from the above section

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

In [203]:

```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Mach
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.datasets import *
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

#https://scikit-learn.org/stable/modules/grid_search.html
parameters = {'n_estimators':[5, 10, 50, 100, 500, 1000], 'eta0': [0.0001, 0.001, 0.01, 0.1]}

clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4, num_boost_round = 10)

model = GridSearchCV(clf, parameters, cv=3, scoring='roc_auc', return_train_score=True, verbose=10, n_jobs=-
model.fit(X_tr_set, y_train)

train_auc =  model.cv_results_['mean_train_score']
train_auc_std = model.cv_results_['std_train_score']
cv_auc = model.cv_results_['mean_test_score']
cv_auc_std = model.cv_results_['std_test_score']
```

```
Fitting 3 folds for each of 24 candidates, totalling 72 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done   2 tasks      | elapsed:    5.8s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:   10.8s
[Parallel(n_jobs=-1)]: Done  16 tasks      | elapsed:   11.9s
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   20.9s
[Parallel(n_jobs=-1)]: Done  34 tasks      | elapsed:   26.1s
[Parallel(n_jobs=-1)]: Done  45 tasks      | elapsed:   32.3s
[Parallel(n_jobs=-1)]: Done  56 tasks      | elapsed:   39.7s
[Parallel(n_jobs=-1)]: Done  65 out of  72 | elapsed:   46.5s remaining:    4.9s
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed:   47.9s finished
```

In [204]:

```
#Results of grid Search
best_params = model.best_params_
print(model.best_score_)
print(model.best_params_)
```

```
0.6754007000120034
{'eta0': 0.0001, 'n_estimators': 5}
```

Plotting Heatmap for HyperParameter vs AUC score

In [189]:

```
model.cv_results_
```

Out[189]:

```
{'mean_fit_time': array([3.35925102, 3.26525625, 4.05664364, 5.05905835, 5.29940192,
        5.16006311, 4.92137321, 5.18205984, 5.02738484, 5.08341026,
        5.35240809, 4.75636069, 4.55102563, 4.18365208, 4.2606647 ,
        4.32665984, 4.24131656, 4.28031325, 4.19500192, 4.08764895,
        4.1413215 , 4.14264369, 4.00597978, 3.57793689]),
 'std_fit_time': array([0.14352081, 0.01806008, 0.79288657, 0.03445995, 0.19585861,
        0.22959996, 0.09953837, 0.09251455, 0.08186816, 0.12784388,
        0.39833902, 0.02195563, 0.1807771 , 0.06604467, 0.0316782 ,
        0.03702761, 0.02268521, 0.05147799, 0.07579975, 0.08463081,
        0.05499708, 0.01552184, 0.20588579, 0.05925685]),
 'mean_score_time': array([0.0853502 , 0.09200724, 0.07167212, 0.07666437, 0.0840083 ,
        0.076334  , 0.09767938, 0.10235357, 0.08133825, 0.11605112,
        0.1006736 , 0.0783387 , 0.08066924, 0.09667802, 0.09866611,
        0.08167307, 0.08733996, 0.09667969, 0.08298874, 0.08034094,
        0.10933272, 0.07767264, 0.08132943, 0.06833879]),
 'std_score_time': array([0.01959422, 0.01525456, 0.00997871, 0.00911558, 0.01699033,
        0.00735758, 0.02359607, 0.0231487 , 0.00952769, 0.05120858,
        0.00611798, 0.01247343, 0.01328009, 0.01879676, 0.03975809,
        0.00758704, 0.00249434, 0.02076832, 0.00704729, 0.01474405,
        0.01686706, 0.01033033, 0.01885537, 0.00579314]),
```

In [190]:

```
#https://github.com/xoelop/Medium-posts/blob/master/3d%20cross%20validation/ML%206%20-%20Gridsearch%20visuliz
#https://qiita.com/bmj0114/items/8009f282c99b77780563
#Saving the obtained results from gridsearch in two dimensional array as dataframe
results = pd.DataFrame(model.cv_results_)
results.head()
```

Out[190]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_eta0	param_n_estimators	params	split0_test_score	sp
0	3.359251	0.143521	0.085350	0.019594	0.0001	5	{'eta0': 0.0001, 'n_estimators': 5}	0.673665	
1	3.265256	0.018060	0.092007	0.015255	0.0001	10	{'eta0': 0.0001, 'n_estimators': 10}	0.673665	
2	4.056644	0.792887	0.071672	0.009979	0.0001	50	{'eta0': 0.0001, 'n_estimators': 50}	0.673665	
3	5.059058	0.034460	0.076664	0.009116	0.0001	100	{'eta0': 0.0001, 'n_estimators': 100}	0.673665	
4	5.299402	0.195859	0.084008	0.016990	0.0001	500	{'eta0': 0.0001, 'n_estimators': 500}	0.673665	

In [191]:

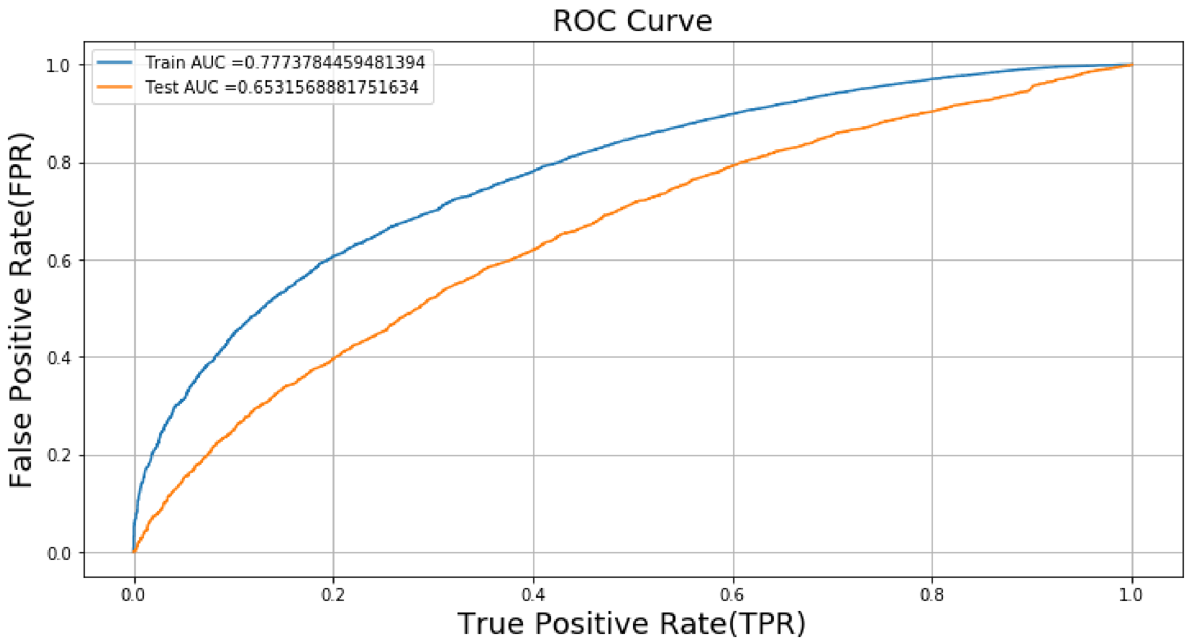
```
def batch_predict(clf, data):

    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 [192]:

```
best_params = {'eta0': 0.0001, 'n_estimators': 5}
```

In [200]:

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

parameters = best_params
clf_one = XGBoostClassifier(**parameters, n_jobs=-1, num_class = 2, num_boost_round = 10)

clf_one.fit(X_tr_set, 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 = clf_one.predict_proba(X_tr_set)[:,1]
y_test_pred = clf_one.predict_proba(X_te_set)[:,1]

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

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)", fontsize = 18)
plt.ylabel("False Positive Rate(FPR)", fontsize = 18)
plt.title("ROC Curve", fontsize = 18)
plt.grid(True)
plt.show()
```



In [194]:

```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [195]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_tpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4949889606193481 for threshold 0.832
[[ 2574   889]
 [ 6341 12641]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4949889606193481 for threshold 0.832
[[1421 1125]
 [4770 9184]]
```

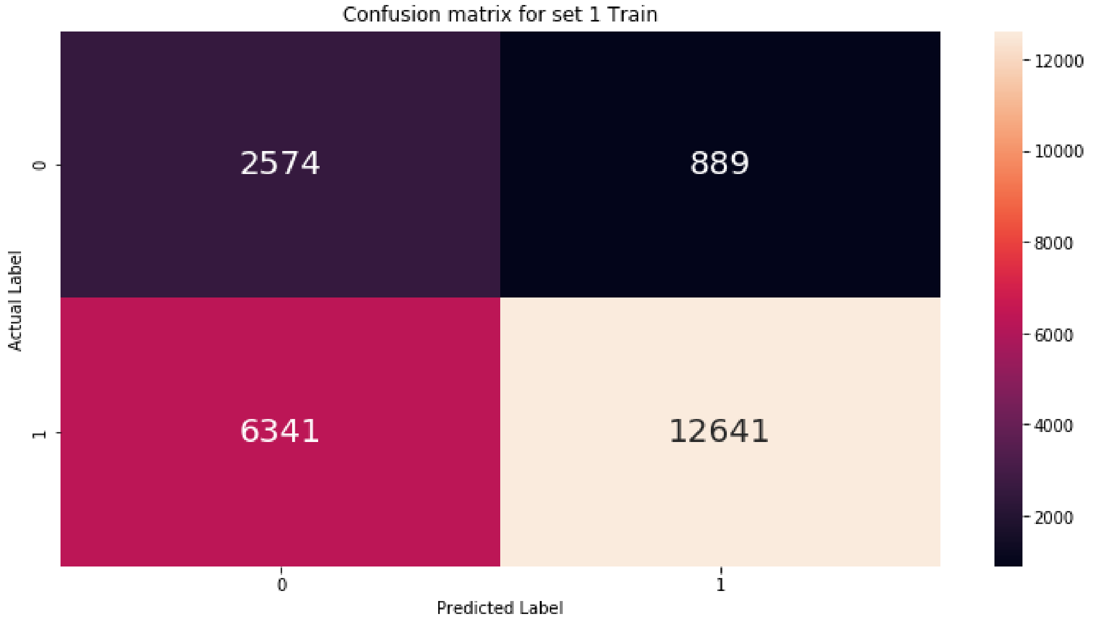
In [196]:

```
conf_mat_set1_train=pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred,tr_thresholds,
train_fpr,train_tpr)),range(2),range(2))
sns.heatmap(conf_mat_set1_train,annot=True,annot_kws={"size":20},fmt='g')
plt.title('Confusion matrix for set 1 Train')
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
```

```
the maximum value of tpr*(1-fpr) 0.4949889606193481 for threshold 0.832
```

Out[196]:

```
Text(87.0, 0.5, 'Actual Label')
```



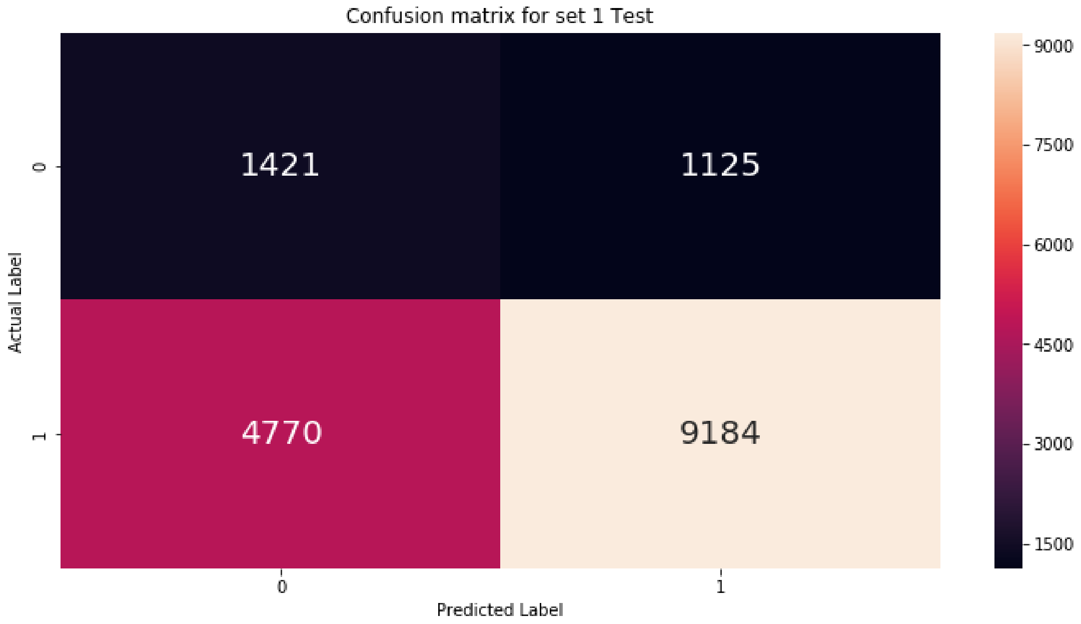
In [197]:

```
conf_mat_set1_test=pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred,tr_thresholds,
train_fpr,train_tpr)),range(2),range(2))
sns.heatmap(conf_mat_set1_test,annot=True,annot_kws={"size":20},fmt='g')
plt.title('Confusion matrix for set 1 Test')
plt.xlabel("Predicted Label")
plt.ylabel("Actual Label")
```

```
the maximum value of tpr*(1-fpr) 0.4949889606193481 for threshold 0.832
```

Out[197]:

```
Text(87.0, 0.5, 'Actual Label')
```



3. Conclusion

In [198]:

```
# Please write down few lines about what you observed from this assignment.
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "N-Estimators", "Learning Rate", "Train AUC", "Test AUC"]
x.add_row(["AVGW2V", "XGBOOST", "5", "0.0001", "0.77", "0.65"])
print(x)
```

```
+------------+---------+--------------+---------------+-----------+----------+
| Vectorizer |  Model  | N-Estimators | Learning Rate | Train AUC | Test AUC |
+------------+---------+--------------+---------------+-----------+----------+
|   AVGW2V   | XGBOOST |      5       |     0.0001    |    0.77   |   0.65   |
+------------+---------+--------------+---------------+-----------+----------+
```