DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** - `Literacy` - `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** - `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: - `nan` - `Dr.` - `Mr.` - `Mrs.` - `Ms.` - `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes

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

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

In [6]:

```
%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
```

```
C:\Users\Public\Anaconda3\lib\site-packages\ge
nsim\utils.py:1197: UserWarning: detected Wind
ows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing ch
unkize to chunkize_serial")
```

1.1 Reading Data

In [7]:

```
project_data = pd.read_csv('C:/Users/pramod reddy chandi/Desk
top/pram/applied ai course/DonorsChoose_2018/train_data.csv')
resource_data = pd.read_csv('C:/Users/pramod reddy chandi/Des
ktop/pram/applied ai course/DonorsChoose_2018/resources.csv')
```

In [8]:

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

```
Number of data points in train data (109248, 1
7)
---------------------------------------------
-----
The attributes of data : ['Unnamed: 0' 'id' 't
eacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_c
ategory'
 'project_subject_categories' 'project_subject
_subcategories'
 'project_title' 'project_essay_1' 'project_es
say_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects
' 'project_is_approved']
```

In [9]:

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

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

Out[9]:

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

In [10]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]
```

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

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	sch
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

In [11]:

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

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

Out[11]:

	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 [33]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inp
lace=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 [34]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

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

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

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

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

1.3 Text preprocessing

In [35]:

```
# 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 [36]:

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

Out[36]:

	Unnamed: 0	id	teach
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b
76127	37728	p043609	3f60494c61921b3b43ab61bdde2

In [37]:

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

In [38]:

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

```
I have been fortunate enough to use the Fairy
Tale STEM kits in my classroom as well as the
STEM journals, which my students really enjoye
d.  I would love to implement more of the Lake
shore STEM kits in my classroom for the next s
chool year as they provide excellent and engag
ing STEM lessons.My students come from a varie
ty of backgrounds, including language and soci
oeconomic status.  Many of them don't have a l
ot of experience in science and engineering an
d these kits give me the materials to provide
these exciting opportunities for my students.E
ach month I try to do several science or STEM/
STEAM projects.  I would use the kits and robo
t to help guide my science instruction in enga
ging and meaningful ways.  I can adapt the kit
s to my current language arts pacing guide whe
re we already teach some of the material in th
e kits like tall tales (Paul Bunyan) or Johnny
 Appleseed.  The following units will be taugh
t in the next school year where I will impleme
```

nt these kits: magnets, motion, sink vs. float, robots. I often get to these units and don't know If I am teaching the right way or using the right materials.   The kits will give me additional ideas, strategies, and lessons to prepare my students in science.It is challenging to develop high quality science activities. These kits give me the materials I need to provide my students with science activities that will go along with the curriculum in my classroom. Although I have some things (like magnets) in my classroom, I don't know how to use them effectively. The kits will provide me with the right amount of materials and show me how to use them in an appropriate way.

==

I teach high school English to students with learning and behavioral disabilities. My students all vary in their ability level. However, the ultimate goal is to increase all students literacy levels. This includes their reading, writing, and communication levels.I teach a really dynamic group of students. However, my students face a lot of challenges. My students all live in poverty and in a dangerous neighborhood. Despite these challenges, I have students who have the the desire to defeat these challenges. My students all have learning disabilities and currently all are performing below grade level. My students are visual learners and will benefit from a classroom that fulfills their preferred learning style.The materials I am requesting will allow my students to be prepared for the classroom with the necessary supplies. Too often I am challenged with students who come to school unprepared for class due t o economic challenges. I want my students to be able to focus on learning and not how they will be able to get school supplies. The supplies will last all year. Students will be able to complete written assignments and maintain a classroom journal. The chart paper will be used to make learning more visual in class and to create posters to aid students in their learning. The students have access to a classroom printer. The toner will be used to print student work that is completed on the classroom Chromebooks.I want to try and remove all barriers for the students learning and create opportunities for learning. One of the biggest barriers is the students not having the resources to get pens, paper, and folders. My students will be able to increase their literacy skills because of this project.

==

\"Life moves pretty fast. If you don't stop and look around once in awhile, you could miss it.\" from the movie, Ferris Bueller's Day Off. Think back...what do you remember about your grandparents? How amazing would it be to be able to flip through a book to see a day in their lives?My second graders are voracious readers! They love to read both fiction and nonfiction books. Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson. They also love to read about insects, space and plants. My students are hungry bookworms! My students are eager to learn and read about the world around them. My kids love to be at school and are like little sponges absorbing everything around them. Their parents work long hours and usually do not see t heir children. My students are usually cared for by their grandparents or a family friend. Most of my students do not have someone who speaks English at home. Thus it is difficult for my students to acquire language.Now think forward... wouldn't it mean a lot to your kids, nieces or nephews or grandchildren, to be able to see a day in your life today 30 years from now? Memories are so precious to us and being able to share these memories with future generations will be a rewarding experience. As part of our social studies curriculum, students will be learning about changes over time. Students will be studying photos to learn about how their community has changed over time. In particular, we will look at photos to study how the land, buildings, clothing, and schools have changed over time. As a culminating activity, my students will capture a slice of their history and preserve it through scrap booking. Key important events in their young lives will be documented with the date, location, and names.   Students will be using photos from home and from school to create their second grade memories.   Their scrap books will preserve their unique stories for future generations to enjoy.Your donation to this project will provide my second graders with an opportunity to learn about social studies in a fun and creative manner. Through their scrapbooks, children will share their story with others and have a historical document for the rest of their lives.

==

\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the biggest enthusiasm for learning. My studen ts learn in many different ways using all of our senses and multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nStudents in my class come from a variety of different backgrounds which makes for wonderful sharing of experiences and cultures, including Native Americans.\r\nOur school is a caring community of successful learners which can be seen through collaborative student project based learning in and out of the classroom. Kindergarteners in my class love to work with hands-on materials and have many different opportunities to practice a skill before it is mastered. Having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the perfect place to learn about agriculture and nutrition. My students love to role play in our pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try cooking with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we learn important math and writing concepts while cooking delicious healthy food for snack time. My students will have a grounded appreciation for the work that went into making the food and knowledge of where the ingredients came from as well as how it's healthy for their bodies. This project would expand our learning of nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroom garden in the spring. We will also create our own cookbooks to be printed and shared with families. \r\nStudents will gain math and literature skills as well as a life long enjoym ent for healthy cooking.nannan

==

My classroom consists of twenty-two amazing sixth graders from different cultures and backgrounds. They are a social bunch who enjoy working in partners and working with groups. They are hard-working and eager to head to middle school next year. My job is to get them ready to make this transition and make it as smooth as possible. In order to do this, my students need to come to school every day and feel safe and ready to learn. Because they are getting ready to head to middle school, I give them lots of choice- choice on where to sit and work, the order to complete assignments, choice of projects, etc. Part of the students feeling safe is the ability for them to come into a welcoming, encouraging environment. My room is colorful and the atmosphere is casual. I want them to take ownership of the classroom because we ALL share it together. Because my time with them is limited, I want to ensure they get the most of this time and enjoy it to the best of their abilities.Currently, we have twenty-two desks of differing sizes, yet the desks are similar to the ones the students will use in middle school. We also have a kidney table with crates for seating. I allow my students to choose their own spots while they are working independently or in groups. More often than not, most of them move out of their desks and onto the crates. Believe it or not, this has proven to be more successful than making them stay at their desks! It is because of this that I am looking toward the “Flexible Seating” option for my classroom.\r\n The students look forward

```
to their work time so they can move around the
 room. I would like to get rid of the constric
ting desks and move toward more “fun” seating 
options. I am requesting various seating so my
 students have more options to sit. Currently,
 I have a stool and a papasan chair I inherite
d from the previous sixth-grade teacher as wel
l as five milk crate seats I made, but I would
 like to give them more options and reduce the
 competition for the “good seats”. I am also r
equesting two rugs as not only more seating op
tions but to make the classroom more welcoming
 and appealing. In order for my students to be
 able to write and complete work without desks
, I am requesting a class set of clipboards. F
inally, due to curriculum that requires groups
 to work together, I am requesting tables that
 we can fold up when we are not using them to 
leave more room for our flexible seating optio
ns.\r\nI know that with more seating options, 
they will be that much more excited about comi
ng to school! Thank you for your support in ma
king my classroom one students will remember f
orever!nannan
==============================================
====
```

In [39]:

```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 [40]:

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

```
\"A person is a person, no matter how small.\"
 (Dr.Seuss) I teach the smallest students with
 the biggest enthusiasm for learning. My stude
nts learn in many different ways using all of
our senses and multiple intelligences. I use a
 wide range of techniques to help all my stude
nts succeed. \r\nStudents in my class come fro
m a variety of different backgrounds which mak
es for wonderful sharing of experiences and cu
ltures, including Native Americans.\r\nOur sch
ool is a caring community of successful learne
rs which can be seen through collaborative stu
dent project based learning in and out of the
classroom. Kindergarteners in my class love to
 work with hands-on materials and have many di
fferent opportunities to practice a skill befo
re it is mastered. Having the social skills to
 work cooperatively with friends is a crucial
aspect of the kindergarten curriculum.Montana
is the perfect place to learn about agricultur
e and nutrition. My students love to role play
```

```
 in our pretend kitchen in the early childhood
 classroom. I have had several kids ask me, \"
Can we try cooking with REAL food?\" I will ta
ke their idea and create \"Common Core Cooking
 Lessons\" where we learn important math and w
riting concepts while cooking delicious health
y food for snack time. My students will have a
 grounded appreciation for the work that went
into making the food and knowledge of where th
e ingredients came from as well as how it is h
ealthy for their bodies. This project would ex
pand our learning of nutrition and agricultura
l cooking recipes by having us peel our own ap
ples to make homemade applesauce, make our own
 bread, and mix up healthy plants from our cla
ssroom garden in the spring. We will also crea
te our own cookbooks to be printed and shared
with families. \r\nStudents will gain math and
 literature skills as well as a life long enjo
yment for healthy cooking.nannan
==============================================
====
```

In [41]:

```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)
```

```
 A person is a person, no matter how small.  (
Dr.Seuss) I teach the smallest students with t
he biggest enthusiasm for learning. My student
s learn in many different ways using all of ou
r senses and multiple intelligences. I use a w
ide range of techniques to help all my student
```

```
s succeed.   Students in my class come from a
variety of different backgrounds which makes f
or wonderful sharing of experiences and cultur
es, including Native Americans.  Our school is
 a caring community of successful learners whi
ch can be seen through collaborative student p
roject based learning in and out of the classr
oom. Kindergarteners in my class love to work 
with hands-on materials and have many differen
t opportunities to practice a skill before it 
is mastered. Having the social skills to work 
cooperatively with friends is a crucial aspect
 of the kindergarten curriculum.Montana is the
 perfect place to learn about agriculture and 
nutrition. My students love to role play in ou
r pretend kitchen in the early childhood class
room. I have had several kids ask me,  Can we 
try cooking with REAL food?  I will take their
 idea and create  Common Core Cooking Lessons 
 where we learn important math and writing con
cepts while cooking delicious healthy food for
 snack time. My students will have a grounded 
appreciation for the work that went into makin
g the food and knowledge of where the ingredie
nts came from as well as how it is healthy for
 their bodies. This project would expand our l
earning of nutrition and agricultural cooking 
recipes by having us peel our own apples to ma
ke homemade applesauce, make our own bread, an
d mix up healthy plants from our classroom gar
den in the spring. We will also create our own
 cookbooks to be printed and shared with famil
ies.   Students will gain math and literature 
skills as well as a life long enjoyment for he
althy cooking.nannan
```

In [42]:

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

```
 A person is a person no matter how small Dr S
euss I teach the smallest students with the bi
ggest enthusiasm for learning My students lear
n in many different ways using all of our sens
es and multiple intelligences I use a wide ran
ge of techniques to help all my students succe
ed Students in my class come from a variety of
 different backgrounds which makes for wonderf
ul sharing of experiences and cultures includi
ng Native Americans Our school is a caring com
munity of successful learners which can be see
n through collaborative student project based 
learning in and out of the classroom Kindergar
teners in my class love to work with hands on 
materials and have many different opportunitie
s to practice a skill before it is mastered Ha
ving the social skills to work cooperatively w
ith friends is a crucial aspect of the kinderg
arten curriculum Montana is the perfect place 
to learn about agriculture and nutrition My st
udents love to role play in our pretend kitche
n in the early childhood classroom I have had 
several kids ask me Can we try cooking with RE
AL food I will take their idea and create Comm
on Core Cooking Lessons where we learn importa
nt math and writing concepts while cooking del
icious healthy food for snack time My students
 will have a grounded appreciation for the wor
k that went into making the food and knowledge
 of where the ingredients came from as well as
 how it is healthy for their bodies This proje
ct would expand our learning of nutrition and 
```

```
agricultural cooking recipes by having us peel
 our own apples to make homemade applesauce ma
ke our own bread and mix up healthy plants fro
m our classroom garden in the spring We will a
lso create our own cookbooks to be printed and
 shared with families Students will gain math
and literature skills as well as a life long e
njoyment for healthy cooking nannan
```

In [43]:

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

In [44]:

```
# 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 stopw
ords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:49<00:00, 2
205.03it/s]
```

In [47]:

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

Out[47]:

```
'a person person no matter small dr seuss i te
ach smallest students biggest enthusiasm learn
ing my students learn many different ways usin
g senses multiple intelligences i use wide ran
```

```
ge techniques help students succeed students c
lass come variety different backgrounds makes 
wonderful sharing experiences cultures includi
ng native americans our school caring communit
y successful learners seen collaborative stude
nt project based learning classroom kindergart
eners class love work hands materials many dif
ferent opportunities practice skill mastered h
aving social skills work cooperatively friends
 crucial aspect kindergarten curriculum montan
a perfect place learn agriculture nutrition my
 students love role play pretend kitchen early
 childhood classroom i several kids ask can tr
y cooking real food i take idea create common 
core cooking lessons learn important math writ
ing concepts cooking delicious healthy food sn
ack time my students grounded appreciation wor
k went making food knowledge ingredients came 
well healthy bodies this project would expand 
learning nutrition agricultural cooking recipe
s us peel apples make homemade applesauce make
 bread mix healthy plants classroom garden spr
ing we also create cookbooks printed shared fa
milies students gain math literature skills we
ll life long enjoyment healthy cooking nannan'
```

In [48]:

```
#Project essay word count

essay_word_count = []

for ess in project_data["essay"] :
    c = len(ess.split())
    essay_word_count.append(c)
```

In [49]:

```
project_data["essay_word_count"] = essay_word_count

project_data['preprocessed_essays'] = preprocessed_essays
```

In [50]:

```
import nltk

from nltk.sentiment.vader import SentimentIntensityAnalyzer

analyser = SentimentIntensityAnalyzer()

pos =[]
neg = []
neu = []
compound = []

for a in tqdm(project_data["preprocessed_essays"]) :
    b = analyser.polarity_scores(a)['neg']
    c = analyser.polarity_scores(a)['pos']
    d = analyser.polarity_scores(a)['neu']
    e = analyser.polarity_scores(a)['compound']
    neg.append(b)
    pos.append(c)
    neu.append(d)
    compound.append(e)
```

```
100%|██████████| 109248/109248 [11:57<00:00, 1
52.21it/s]
```

In [51]:

```
project_data["pos"] = pos
project_data["neg"] = neg
project_data["neu"] = neu
project_data["compound"] = compound
```

1.4 Preprocessing of $project_{t}it \leq$

In [52]:

```
# similarly you can preprocess the titles also
project_data.columns
```

Out[52]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
her_prefix', 'school_state',
       'Date', 'project_grade_category', 'proj
ect_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', '
project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'clean_categories', 'clean_subcategorie
s', 'essay', 'essay_word_count',
       'preprocessed_essays', 'pos', 'neg', 'n
eu', 'compound'],
      dtype='object')
```

In [53]:

```
#sent1= decontracted(project_data['project_title'].values[200
00])
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e not in sto
pwords)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:02<00:00, 4
6464.23it/s]
```

In [54]:

```
#Project title word count
title_word_count = []

for a in project_data["project_title"] :
    b = len(a.split())
    title_word_count.append(b)

project_data["title_word_count"] = title_word_count
```

In [55]:

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

1.5 Preparing data for models

In [56]:

```
project_data.columns
```

Out[56]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
her_prefix', 'school_state',
       'Date', 'project_grade_category', 'proj
ect_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', '
project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'clean_categories', 'clean_subcategorie
s', 'essay', 'essay_word_count',
       'preprocessed_essays', 'pos', 'neg', 'n
eu', 'compound',
       'title_word_count', 'preprocessed_title
'],
      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
```

In [12]:

```
Y=project_data['project_is_approved']
```

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]:

```
project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_title'] = preprocessed_title
```

In [60]:

```
column_values=['clean_categories', 'clean_subcategories', 'school_state', 'project_grade_category', 'teacher_prefix','preprocessed_essays','preprocessed_title' ,'price','quantity','teacher_number_of_previously_posted_projects','pos','neg','neu','compound','title_word_count','essay_word_count']

def select_columns(dataframe, column_names):
    new_frame = dataframe.loc[:, column_names]
    return new_frame

process_columns=select_columns(project_data,column_values)
```

In [61]:

```
process_columns.head()
```

Out[61]:

	clean_categories	clean_subcategories	school_state	project_grade_category
0	Math_Science	AppliedSciences Health_LifeScience	CA	Grades PreK-2
1	SpecialNeeds	SpecialNeeds	UT	Grades 3-5
2	Literacy_Language	Literacy	CA	Grades PreK-2
3	AppliedLearning	EarlyDevelopment	GA	Grades PreK-2
4	Literacy_Language	Literacy	WA	Grades 3-5

1.5.1 Vectorizing Categorical data

In [62]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_categories= CountVectorizer(vocabulary=list(sorted
_cat_dict.keys()), lowercase=False, binary=True)

vectorizer_categories.fit(process_columns['clean_categories'].
values)

categories_one_hot = vectorizer_categories.transform(process_
columns['clean_categories'].values)
```

```
print(vectorizer_categories.get_feature_names())

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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'M
usic_Arts', 'AppliedLearning', 'SpecialNeeds',
 'Health_Sports', 'Math_Science', 'Literacy_La
nguage']
Shape of  matrix after one hot encodig  (10924
8, 9)
```

In [63]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_subcategories = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_subcategories.fit(process_columns['clean_subcategories'].values)

print(vectorizer_subcategories.get_feature_names())

sub_categories_one_hot = vectorizer_subcategories.transform(process_columns['clean_subcategories'].values)

print("Shape of  matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLi
teracy', 'ParentInvolvement', 'Extracurricular
', 'Civics_Government', 'ForeignLanguages', 'N
utritionEducation', 'Warmth', 'Care_Hunger', '
SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_Car
eerPrep', 'Music', 'History_Geography', 'Healt
h_LifeScience', 'EarlyDevelopment', 'ESL', 'Gy
m_Fitness', 'EnvironmentalScience', 'VisualArt
s', 'Health_Wellness', 'AppliedSciences', 'Spe
cialNeeds', 'Literature_Writing', 'Mathematics
', 'Literacy']
Shape of  matrix after one hot encodig  (10924
8, 30)
```

In [64]:

```
# we use count vectorizer to convert the values of categorica
l data :school_state
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_schoolstate= CountVectorizer()
vectorizer_schoolstate.fit(process_columns['school_state'])

print(vectorizer_schoolstate.get_feature_names())

school_state_one_hot = vectorizer_schoolstate.transform(proce
ss_columns['school_state'].values)

print("Shape of  matrix after one hot encodig ",school_state_
one_hot.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc
', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', '
in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi',
 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh
', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', '
pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va',
 'vt', 'wa', 'wi', 'wv', 'wy']
Shape of  matrix after one hot encodig  (10924
8, 51)
```

In [65]:

```
#we use count vectorizer to convert the values of categorical data :project_grade_category
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_project_grade_category = CountVectorizer(stop_words=None)

k=process_columns['project_grade_category']



k.replace(['Grades PreK-2', 'Grades 6-8', 'Grades 3-5','Grades 9-12'], ['A1', 'B2' ,'C3', 'D4'],inplace=True)

vectorizer_project_grade_category.fit(k)

project_grade_category_one_hot=vectorizer_project_grade_category.transform(process_columns['project_grade_category'].values)

print("Shape of matrix after one hot encodig ",project_grade_category_one_hot.shape)
```

```
Shape of matrix after one hot encodig  (109248, 4)
```

In [66]:

```
#we use count vectorizer to convert the values of categorical data : teacher_prefix
# getting error as we have null balues replacing them with 0
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_teacher_prefix = CountVectorizer()
project_data['teacher_prefix'].unique()

process_columns['teacher_prefix'].fillna("", inplace = True)
```

```
vectorizer_teacher_prefix.fit(process_columns['teacher_prefix'
].values)
print(vectorizer_teacher_prefix.get_feature_names())

teacher_prefix_one_hot = vectorizer_teacher_prefix.transform(
process_columns['teacher_prefix'].values)

print("Shape of  matrix after one hot encodig ",teacher_prefi
x_one_hot.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of  matrix after one hot encodig  (10924
8, 5)
```

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

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [67]:

```
# We are considering only the words which appeared in at leas
t 10 documents(rows or projects).
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_bow_essay = CountVectorizer(min_df=10,max_features
=250)
vectorizer_bow_essay.fit(process_columns['preprocessed_essays'
])

text_bow= vectorizer_bow_essay.transform(process_columns['pre
processed_essays'])

print("Shape of  matrix after one hot encodig ",text_bow.shape)
```

```
Shape of  matrix after one hot encodig  (109248, 250)
```

In [68]:

```
# before you vectorize the title make sure you preprocess it
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit(process_columns['preprocessed_title'])

title_bow = vectorizer_bow_title.transform(process_columns['preprocessed_title'])

print("Shape of matrix after one hot encodig title_bow",title_bow.shape)
```

```
Shape of matrix after one hot encodig title_bow (109248, 91)
```

1.5.3 Vectorizing Numerical features

In [69]:

```
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 [70]:

```
#scaling of price feature

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

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


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

print(price_standardized.shape)
```

```
(109248, 1)
```

In [71]:

```
#scaling of qunatity feature

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

```
# 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)

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



# Now standardize the data with above maen and variance.
quantity_standardized= quantity_scalar.transform(process_columns['quantity'].values.reshape(-1, 1))

print(quantity_standardized.shape)
```

```
(109248, 1)
```

In [72]:

```
#scaling of teachers number of previously posted projects

from sklearn.preprocessing import Normalizer

normalizer_projects_num = Normalizer()

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

```
normalizer_projects_num.fit(process_columns['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

prev_projects = normalizer_projects_num.transform(process_columns['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print(prev_projects.shape)
```

```
(109248, 1)
```

In [73]:

```
# normalixing the title word count

from sklearn.preprocessing import Normalizer

normalizer_title_word = Normalizer()

normalizer_title_word.fit(process_columns['title_word_count'].values.reshape(-1,1))

title_word_count = normalizer_title_word.transform(process_columns['title_word_count'].values.reshape(-1,1))

print(title_word_count.shape)
print("="*100)
```

```
(109248, 1)
==============================================
==============================================
========
```

In [74]:

```
# normalixing the essay word count

from sklearn.preprocessing import Normalizer
```

```
normalizer_ess_count = Normalizer()

normalizer_ess_count.fit(process_columns['essay_word_count'].values.reshape(-1,1))

essay_word_count = normalizer_ess_count.transform(process_columns['essay_word_count'].values.reshape(-1,1))

print(essay_word_count.shape)
```

```
(109248, 1)
```

In [75]:

```
#normalizing the data for  essay sentiment-pos
from sklearn.preprocessing import Normalizer
normalizer_pos = Normalizer()

normalizer_pos.fit(process_columns['pos'].values.reshape(-1,1))

essay_sent_pos = normalizer_pos.transform(process_columns['pos'].values.reshape(-1,1))

print(essay_sent_pos.shape)
```

```
(109248, 1)
```

In [76]:

```
#normalizing the data for  essay sentiment-neg
from sklearn.preprocessing import Normalizer

normalizer_neg= Normalizer()

normalizer_neg.fit(process_columns['neg'].values.reshape(-1,1))
```

```
essay_sent_neg = normalizer_neg.transform(process_columns['neg'].values.reshape(-1,1))

print(essay_sent_neg.shape)
```

```
(109248, 1)
```

In [77]:

```
#normalizing the data for  essay sentiment-neu
from sklearn.preprocessing import Normalizer

normalizer_nue= Normalizer()

normalizer_nue.fit(process_columns['neu'].values.reshape(-1,1))

essay_sent_nue = normalizer_nue.transform(process_columns['neu'].values.reshape(-1,1))

print(essay_sent_nue.shape)
```

```
(109248, 1)
```

In [78]:

```
#normalizing the data for  essay sentiment-compound
from sklearn.preprocessing import Normalizer

normalizer_compound= Normalizer()

normalizer_compound.fit(process_columns['compound'].values.reshape(-1,1))

essay_sent_comp = normalizer_compound.transform(process_columns['compound'].values.reshape(-1,1))
```

```
print(essay_sent_comp.shape)
print("="*100)
```

```
(109248, 1)
==============================================
==============================================
========
```

1.5.4 Merging all the above features

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

In [79]:

```
from scipy.sparse import hstack

#define categorical and numerical features
cat_num=hstack((school_state_one_hot,categories_one_hot,sub_categories_one_hot,teacher_prefix_one_hot,project_grade_category_one_hot,price_standardized, quantity_standardized, prev_projects, title_word_count, essay_word_count, essay_sent_pos, essay_sent_neg, essay_sent_nue, essay_sent_comp))
```

In [80]:

```
type(preprocessed_essays)
```

Out[80]:

```
list
```

In [81]:

```
type(preprocessed_title)
```

Out[81]:

```
list
```

In [82]:

```
#concatinate essay and title
essay_text=preprocessed_essays + preprocessed_title
```

In [83]:

```
len(essay_text)
```

Out[83]:

```
218496
```

In [84]:

```
#Load the review text corpus.
X_train = essay_text

#Initializing the TF-IDF constructor
tf_idf_obj = TfidfVectorizer(max_features=2000,stop_words=stopwords, use_idf=True, ngram_range=(1,1), dtype='float64').fit(X_train)

X_vectors = tf_idf_obj.transform(X_train)

print("Total number of unique features(words) present in the corpus: ",len(tf_idf_obj.get_feature_names()))
```

```
Total number of unique features(words) present
 in the corpus:  2000
```

In [85]:

```
X_vectors.shape
```

Out[85]:

```
(218496, 2000)
```

In [86]:

```
#I have used the code from: https://buhrmann.github.io/tfidf-analysis.html
def get_top_features(tf_idf_obj, feat_names, top_n_features):
    # returns a dataframe with top n features :
    feat_index = np.argsort(tf_idf_obj.idf_)
    top_features = [(feat_names[i], tf_idf_obj.idf_[i]) for i in feat_index[:top_n_features]]
    df_topFeatures = pd.DataFrame(data=top_features, columns = ['Top Words','TF-IDF Value'])
    return df_topFeatures

#Get the mean TF-IDF Scores
tfidf_mean = np.mean(X_vectors, axis = 0)
tfidf_mean = np.array(tfidf_mean)[0].tolist()

#List of all the feature names
feat_names = tf_idf_obj.get_feature_names()

#Get e DataFrame containing the top TFIDF words along with their scores.
top_n_features = 2000
top_features = get_top_features(tf_idf_obj, feat_names, top_n_features)

#Print the top 10 features.
#Words are arranged in increasing order of idf_ scores. Remember, less idf_(inverse) scores means more important the document.
top_features.head(10)
```

Out[86]:

	Top Words	TF-IDF Value
0	students	1.003847
1	nannan	1.022433

2	school	1.077401
3	learning	1.165487
4	not	1.199571
5	learn	1.204691
6	many	1.247720
7	come	1.300900
8	love	1.336404
9	also	1.341459

In [87]:

```
top_features.shape
```

Out[87]:

```
(2000, 2)
```

In [88]:

```
X_vectors
```

Out[88]:

```
<218496x2000 sparse matrix of type '<class 'numpy.float64'>'
	with 18852550 stored elements in Compressed Sparse Row format>
```

In [87]:

```
#Generate the Co-Occurence Matrix
def get_coOccuranceMatrix(X_train, top_features, window): #window = 2 means 2 on either side of the given word. Lets look at an example. data = [a,b,c,d,e,f,g,h],
    print("Generating the Co Occurence Matrix....")        #if window = 2, for letter c, we will have neighborhood = [a,b,c,d,e]. for f -> [d,e,f,g,h]
    dim=top_features.shape[0]
    square_matrix = np.zeros((dim,dim),int)

    values = [i for i in range(0,top_features.shape[0])]  #Contains all the top TF-IDF Scores as values.
    keys = [str(i) for i in top_features['Top Words']]    #Contains all the corresponding features names as keys.
    lookup_dict = dict(zip(keys,values))                  #We will use this dictionary as a look up table

    top_words= keys

    #Processing each reviews to build the co-occurence Matrix
    for reviews in tqdm(X_train):
        #Split each review into words
        words = reviews.split()
        lnt = len(words)
        for i in range(0,len(words),1):
            idx_of_neigbors= []
            if((i-window >= 0) and (i+window < lnt)):
                idx_of_neigbors = np.arange(i-window,i+window+1)
            elif((i-window < 0) and (i+window < lnt)):
                idx_of_neigbors = np.arange(0, i+window+1)
            elif((i-window >= 0) and (i+window >= lnt)):
                idx_of_neigbors = np.arange(i-window, lnt)
            else:
                pass
            #nei = [words[x] for x in idx_of_neigbors]
            #print(words[i],"---------",nei)
            #print(idx_of_neigbors)

            for j in idx_of_neigbors:
                if((words[j] in top_words) and (words[i] in top_words)):
                    row_idx = lookup_dict[words[i]]     #Get the index of the ith word from the lookup table
                    col_idx = lookup_dict[words[j]]     #Get
```

```
the index of the jth word from the lookup table
                    square_matrix[row_idx,col_idx] += 1 #If w
ord[i] and word[j] occurs in a neighbourhood of 5, there co o
ccurence will be increases by one.
                else:
                    pass

    #Fill all the diagonal elements of the co-occurence matri
x with 0, as co-occurence of a word with itlself is always ze
ro.
    np.fill_diagonal(square_matrix, 0)
    print("Co Occurence Matrix is generated....")

    #Create a co-occurence dataframe.
    co_occur_df=pd.DataFrame(data=square_matrix, index=keys,
columns=keys)
    return co_occur_df

co_occur_matrix = get_coOccuranceMatrix(X_train, top_features
, window=5)
```

```
Generating the Co Occurence Matrix....
```

```
100%|██████████| 218496/218496 [58:45<00:00, 6
1.98it/s]
```

```
Co Occurence Matrix is generated....
```

In [88]:

```
co_occur_matrix.shape
```

Out[88]:

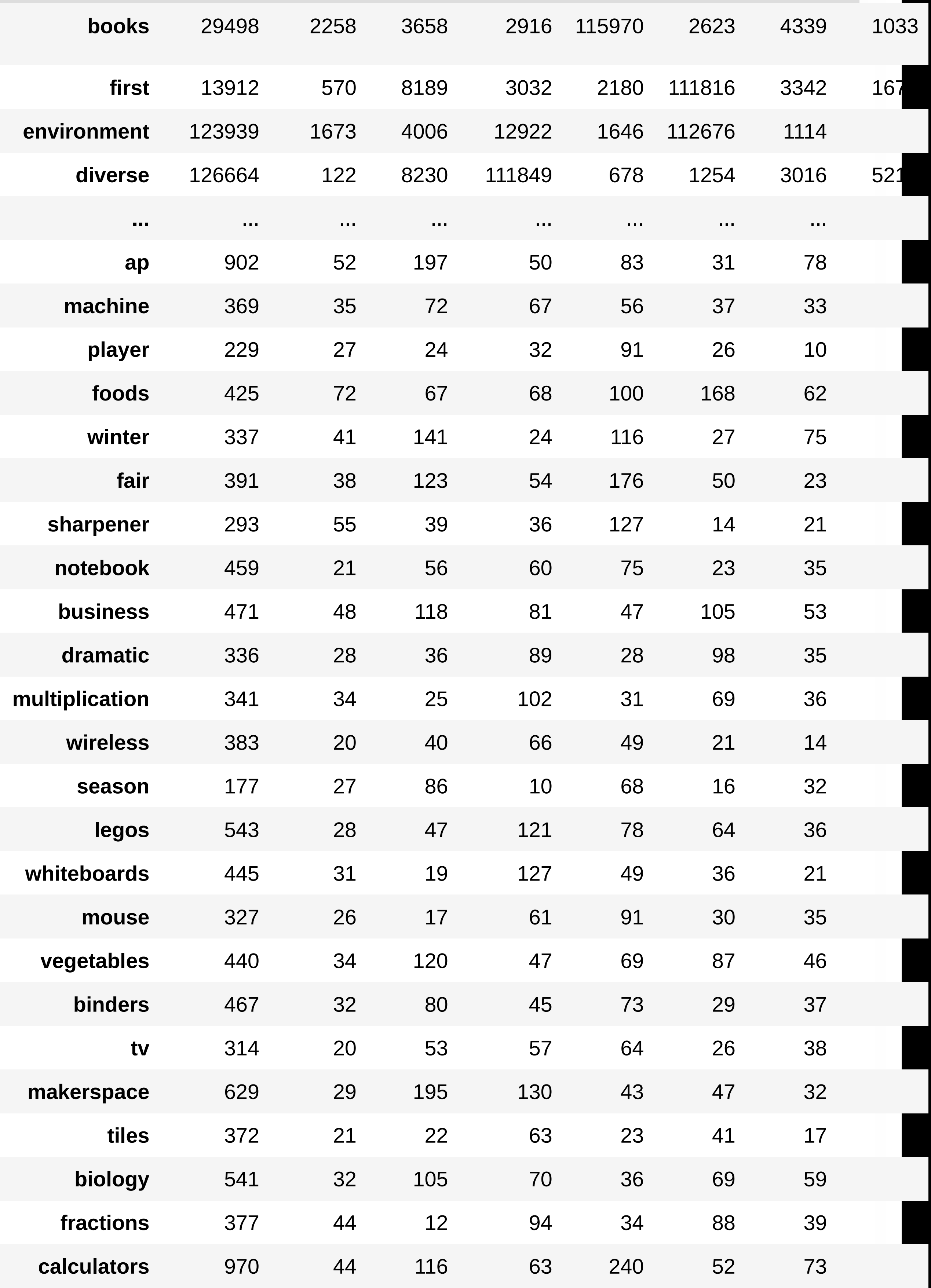
```
(2000, 2000)
```

In [89]:

```
co_occur_matrix
```

Out[89]:

	students	nannan	school	learning	not	learn	many	com
students	0	17575	125991	203705	62174	292380	178925	52309
nannan	17575	0	4210	10182	2895	4294	1427	156
school	125991	4210	0	16725	20583	130272	23246	137022
learning	203705	10182	16725	0	10813	9810	9094	493
not	62174	2895	20583	10813	0	8436	126451	115801
learn	292380	4294	130272	9810	8436	0	6930	12187
many	178925	1427	23246	9094	126451	6930	0	123716
come	52309	1563	137022	4932	115801	121871	123716	
love	261435	2283	120678	126442	4027	123222	3638	3028
also	28984	1227	5970	7210	7159	114010	3295	140
day	34272	2372	130063	116962	5132	12073	4203	117971
class	143080	2417	6994	115605	5175	4998	3911	427
want	250265	1136	8219	7287	5820	120441	2566	2290
reading	371906	4480	4911	6385	6254	113279	4802	145
allow	30966	1244	1784	6219	3099	3212	1059	
provide	134928	1273	5413	8799	113550	111956	3385	
teach	18738	531	11378	2600	2632	3535	1663	1134
high	23289	532	25813	1953	2611	1373	3408	284
way	126455	2229	3117	5422	3206	6127	1292	
world	19721	2407	112789	4140	2686	223445	2220	11092
materials	133912	1791	3877	7285	4821	3746	2198	
best	129928	1395	5616	5866	2668	9045	1962	149
learners	129346	1479	4652	4039	1557	2505	3898	1930
children	8127	1122	7370	5184	4239	5579	4423	272
using	15588	1020	1459	3898	1752	2883	1149	
eager	17750	185	7061	2329	922	127297	1937	11400



books	29498	2258	3658	2916	115970	2623	4339	1033
first	13912	570	8189	3032	2180	111816	3342	167
environment	123939	1673	4006	12922	1646	112676	1114	
diverse	126664	122	8230	111849	678	1254	3016	521
...	...	...	...	...	...	...	...	
ap	902	52	197	50	83	31	78	
machine	369	35	72	67	56	37	33	
player	229	27	24	32	91	26	10	
foods	425	72	67	68	100	168	62	
winter	337	41	141	24	116	27	75	
fair	391	38	123	54	176	50	23	
sharpener	293	55	39	36	127	14	21	
notebook	459	21	56	60	75	23	35	
business	471	48	118	81	47	105	53	
dramatic	336	28	36	89	28	98	35	
multiplication	341	34	25	102	31	69	36	
wireless	383	20	40	66	49	21	14	
season	177	27	86	10	68	16	32	
legos	543	28	47	121	78	64	36	
whiteboards	445	31	19	127	49	36	21	
mouse	327	26	17	61	91	30	35	
vegetables	440	34	120	47	69	87	46	
binders	467	32	80	45	73	29	37	
tv	314	20	53	57	64	26	38	
makerspace	629	29	195	130	43	47	32	
tiles	372	21	22	63	23	41	17	
biology	541	32	105	70	36	69	59	
fractions	377	44	12	94	34	88	39	
calculators	970	44	116	63	240	52	73	

sand	277	24	10	42	48	37	27
cooking	354	35	75	62	60	104	43
clay	337	22	44	52	72	41	29
chemistry	462	25	86	66	40	82	39
kindles	506	33	43	112	76	34	36
dash	442	16	27	102	26	98	11

2000 rows × 2000 columns

In [90]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler(with_mean=True).fit(co_occur_matrix)
df_standardize = scalar.transform(co_occur_matrix)
```

```
C:\Users\Public\Anaconda3\lib\site-packages\sk
learn\preprocessing\data.py:645: DataConversio
nWarning:

Data with input dtype int32 were all converted
 to float64 by StandardScaler.

C:\Users\Public\Anaconda3\lib\site-packages\ip
ykernel_launcher.py:3: DataConversionWarning:

Data with input dtype int32 were all converted
 to float64 by StandardScaler.
```

In [91]:

```
type(df_standardize)
```

Out[91]:

```
numpy.ndarray
```

In [93]:

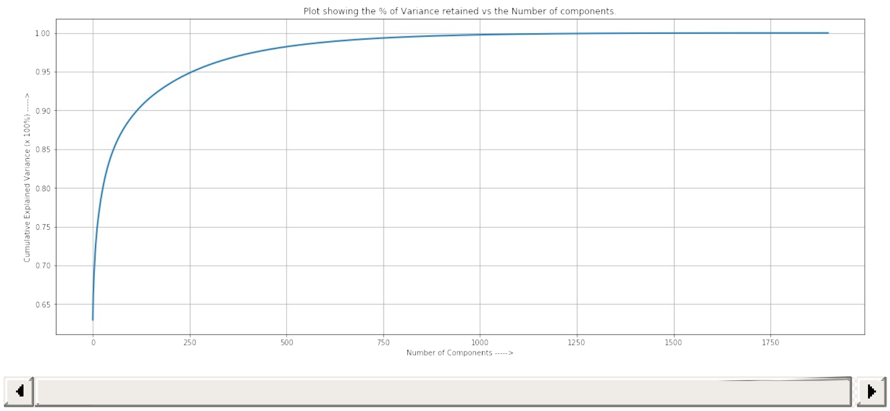
```
df_standardize.shape
```

Out[93]:

```
(2000, 2000)
```

In [94]:

```
from sklearn.decomposition import TruncatedSVD

#Inititalize the truncated SVD object.
svd = TruncatedSVD(n_components=1900,
                   algorithm='randomized',
                   n_iter=10,
                   random_state=0)
data=svd.fit_transform(df_standardize)

cum_var_explained = np.cumsum(svd.explained_variance_ratio_)

# Plot the SVD spectrum
plt.figure(1, figsize=(20, 8))
plt.plot(cum_var_explained, linewidth=2)
plt.axis('tight')
plt.grid()
plt.title('Plot showing the % of Variance retained vs the Number of components.')
plt.xlabel('Number of Components ----->')
plt.ylabel('Cumulative Explained Variance (x 100%) ----->')
plt.show()
```


Plot showing the % of Variance retained vs the Number of components.
Cumulative Explained Variance (x 100%) ----->
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0
250
500
750
1000
1250
1500
1750
Number of Components ----->

from the figue we can observe that the 250 features explain more than 95% variance of data

In [95]:

```
# final matrix with 250 features
from sklearn.decomposition import TruncatedSVD

#Inititalize the truncated SVD object.
svd = TruncatedSVD(n_components=250,
                   algorithm='randomized',
                   n_iter=10,
                   random_state=0)
matrix_final=svd.fit_transform(df_standardize)
```

In [96]:

```
matrix_final.shape
```

Out[96]:

```
(2000, 250)
```

In [194]:

```
matrix_final=np.transpose(matrix_final)
```

In [195]:

```
matrix_final.shape
```

Out[195]:

```
(250, 2000)
```

In [217]:

```
import pickle
f=open('11svd_matrix.pckl','wb')
pickle.dump([matrix_final],f)
f.close()
```

In [89]:

```
import pickle as pickle
#with open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai course/DonorsChoose_2018/cat_num.pckl', 'rb') as f:
f=open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai course/DonorsChoose_2018/11svd_matrix.pckl','rb')
matrix_final=pickle.load(f)
f.close()
```

In [90]:

```
len(matrix_final)
```

Out[90]:

```
1
```

In [91]:

```
data=[]
for x in matrix_final:
    data.extend(x)
```

In [92]:

```
import pandas as pd
df = pd.DataFrame(data)
```

In [93]:

```
df.shape
```

Out[93]:

```
(250, 2000)
```

In [66]:

```
i=0
list_of_sentance_train=[]
for sentance in process_columns['preprocessed_essays']:
    list_of_sentance_train.append(sentance.split())
```

In [94]:

```
w2v_words = tf_idf_obj.get_feature_names()
```

In [68]:

```
process_columns['preprocessed_essays']
```

Out[68]:

```
0          i fortunate enough use fairy tale st
em kits cl...
1          imagine 8 9 years old you third grad
e classroo...
2          having class 24 students comes diver
se learner...
3          i recently read article giving stude
nts choice...
4          my students crave challenge eat obst
acles brea...
5          it end school year routines run cour
se student...
6          sitting still overrated it makes sen
se opera m...
7          it not enough read book write essay 
connect de...
8          never society rapidly changed techno
logy invad...
9          do remember first time saw star wars
 wall e ro...
```

```
10        my students yearn classroom environm
ent matche...
11        media cinematography extremely fast 
growing su...
12        computer coding robotics second grad
ers excite...
13        i teach 4th grade math writing socia
l studies ...
14        in classroom explore delve real worl
d problems...
15        a typical day classroom starts 30 mi
nutes scho...
16        we love technology in classroom tech
nology enh...
17        teachers love teaching teach childre
n love lea...
18        do remember book read made fall love
 reading t...
19        my students learning become lovers r
eading i r...
20        throughout school year i hope enable
 students ...
21        children spend much much time connec
ted media ...
22        education nurturing justice engaging
 students ...
23        everyday students interact technolog
y enhance ...
24        i teach six amazing children autism 
each day m...
25        everyday students excited come show 
books prac...
26        i half day pre k i two sets students
 benefit m...
27        our school urban public school serve
s students...
28        each day i teach class every grade i
```

```
 never bor...
29        my students struggling readers i sup
port stude...
                                ...

109218    i work school approximately 800 stud
ents all b...
109219    as new teacher working high poverty 
district i...
109220    love sing create move learn you come
 right pla...
109221    our school 95 percent free reduced l
unch some ...
109222    my students live oklahoma city oklah
oma grades...
109223    my 6th period class consist 36 stude
nts variet...
109224    i work wonderful group second grader
s energeti...
109225    as teacher librarian i get share lov
e written ...
109226    i incredibly lucky spend days 18 awe
some diver...
109227    i teach fifth graders low income hig
h poverty ...
109228    every child deserves champion adult 
never give...
109229    my students low income families arou
nd small n...
109230    our school encountered great loss du
e devastat...
109231    motivated learn students never cease
 amaze cur...
109232    although physical education mandated
 one hundr...
109233    my students excited happy frustrated
 sad angry...
```

```
109234    my first graders creative innovative
 talented ...
109235    my classroom revolving door they eag
er learner...
109236    my school work microsoft teals progr
am offer i...
109237    each day students eagerly anticipate
 read alou...
109238    i teach 17 amazing students title on
e school m...
109239    my students often worry things outsi
de educati...
109240    our students come multiple different
 backgroun...
109241    my students year love science engine
ering ever...
109242    i teach first grade title i school a
lthough st...
109243    our day starts 100 students athletes
 low incom...
109244    my students range age four five year
s old atte...
109245    we title 1 school 650 total students
 our eleme...
109246    i teach many different types student
s my class...
109247    my first graders eager learn world a
round they...
Name: preprocessed_essays, Length: 109248, dty
pe: object
```

In [69]:

```
# average Word2Vec of essays
# compute average word2vec for each review.
essay_vectors= []; # the avg-w2v for each sentence/review is
stored in this list
```

```
for sent in tqdm(list_of_sentance_train): # for each review/s
entence
    sent_vec = np.zeros(250) # as word vectors are of zero le
ngth 50, you might need to change this to 300 if you use goog
le's w2v
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            i=top_features[top_features['Top Words'] == word]
.index.tolist()
            vec=df.iloc[:,i]
            sent_vec += vec.values.reshape(250)
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    essay_vectors.append(sent_vec)
essay_vectors= np.array(essay_vectors)
print(essay_vectors.shape)
print(essay_vectors[0])
```

```
100%|██████████| 109248/109248 [3:21:20<00:00,
  9.04it/s]
```

```
(109248, 250)
[ 8.82182995e+01 -1.99121171e+00  2.84858660e+
00 -2.15640438e+00
  1.80422733e+00  2.99075273e+00  2.00473324e+
00 -1.24707457e+00
  3.51026002e+00  1.24056233e+00 -2.90918296e-
01  1.17199389e+00
  9.06822087e-01  3.84162269e-01 -1.03189689e+
00 -1.24811246e+00
  1.23276930e-01 -8.61289627e-01 -2.37323124e-
01 -4.58108579e-01
  7.65585692e-01  1.76208715e+00 -1.52329752e-
01 -5.23635093e-01
  1.15065319e+00  3.57658697e-01 -1.51911066e+
00  1.07641284e+00
  2.08937415e+00 -2.35295416e-01 -9.40530325e-
01  2.32541636e-01
  7.57728473e-01 -2.42344434e-01  1.11565270e+
00 -1.16320725e+00
  1.00816289e-01  6.14247852e-01 -6.65203872e-
01 -1.88321750e-01
  4.08783630e-01  4.85075833e-01 -7.48511067e-
02  8.42895663e-01
  1.04649510e-01  4.26879711e-01 -1.59735150e-
01 -2.19732387e-01
 -4.85497774e-01 -2.67124252e-01 -2.97112612e-
01  1.62239920e-01
  8.59167317e-01  8.54602290e-01  2.60146696e-
01  3.71522493e-01
  8.95260350e-01  5.47689737e-01 -2.10744675e-
01 -2.30370648e-01
 -7.09462055e-01  2.21364001e-01 -1.23042135e+
00 -1.60918760e-01
 -9.14788269e-02  6.41718368e-01 -8.25137745e-
01  4.62582065e-02
 -1.22252529e-01 -1.03681762e-01  3.66694231e-
01 -1.48937831e-01
  4.03496365e-02  1.71372540e-01  1.31698938e-
01  3.03210321e-01
 -2.51369583e-01 -3.05607195e-01  4.30521367e-
01 -1.07371522e-01
  6.06541091e-01  3.39207399e-01 -2.43949012e-
01  5.79706501e-01
  3.32966659e-02 -2.73261155e-01 -2.43407615e-
01  2.22966754e-01
  1.44222505e-01  7.70810512e-02  1.62629227e-
01  9.00603557e-02
 -5.00819942e-02 -1.92670533e-01  2.01751364e-
01  2.93819369e-03
 -1.11324869e-01 -1.82533919e-01 -2.30147050e-
02 -1.17355143e-01
```

```
  5.19203911e-01  2.52323525e-02  1.70897421e-
01 -3.50618820e-02
  3.22390088e-01 -1.43103036e-01  1.25361482e-
01  1.25006288e-01
  5.43532002e-02 -1.10931252e-01  1.66317189e-
01  1.87551087e-01
 -4.57559343e-02 -2.34225112e-01 -1.56933812e-
01 -1.61374268e-01
 -1.31758975e-01 -1.37372039e-01  8.47785956e-
02 -1.65717385e-01
  4.44009604e-02 -8.00200909e-02  1.89696426e-
01 -2.10049389e-01
 -9.40382219e-02 -5.64526074e-02 -9.80084920e-
02  1.38126907e-01
  2.33979995e-01  1.67923708e-01 -2.16832831e-
01  2.43729305e-01
 -1.53856062e-01 -1.01019781e-01  2.65840615e-
01 -3.27760626e-02
  1.08715434e-01 -1.16403402e-01 -3.54565277e-
02 -1.95252992e-01
 -6.86191511e-02 -4.22436827e-01  6.12067092e-
02 -3.43070916e-01
 -2.34170844e-02  2.72126797e-01 -6.28538892e-
03 -1.75993186e-01
 -2.42892641e-01 -4.82767840e-02  1.17690604e-
01 -2.57061106e-01
 -1.52770030e-01  2.81257810e-02 -7.48589882e-
02  8.38068373e-02
 -5.69747520e-03  2.81045914e-03  4.63559488e-
02  2.79461773e-01
 -3.61036413e-02  4.49087401e-02  1.83305089e-
01 -1.65104765e-01
  9.43403260e-02  5.34487069e-02  1.93189977e-
01 -9.56145762e-02
 -2.40516907e-01  2.56261557e-02  4.43608626e-
02 -4.22367542e-02
  1.45040077e-01  2.14349321e-01  2.79275970e-
```

```
02 -1.15782065e-01
 -1.11824739e-01  1.18551338e-01 -4.26270907e-
02 -2.03517308e-01
 -4.84462638e-02  6.63220481e-02 -4.65099620e-
02 -2.81490543e-01
  6.25469164e-02  5.18597283e-03 -9.16834237e-
02  1.16880945e-01
 -1.06061894e-01 -5.07783568e-02 -1.21879617e-
01 -1.53042474e-01
 -1.05476186e-01 -5.28973204e-02  1.00308068e-
01  1.16776921e-01
 -6.94094860e-02 -2.90910726e-02  8.85088961e-
02 -6.80391314e-02
 -8.58971728e-02  9.49456734e-02  1.96296055e-
01 -2.45916836e-02
  2.02832245e-01  4.55597380e-02 -1.14192358e-
01 -5.31505813e-02
  9.25931712e-02 -1.50875455e-01  6.70998311e-
02 -6.12869247e-02
  3.30133797e-02 -2.75103514e-02 -2.59339251e-
02  7.99772093e-02
 -5.22357020e-02 -1.73190571e-01  8.48446010e-
02  1.20879986e-01
 -1.08058899e-01 -2.91046934e-02 -1.82466524e-
01 -3.94250814e-02
  6.51146990e-02  5.16948048e-02 -8.98473164e-
02 -8.38617372e-02
 -9.88492301e-02 -7.90733876e-02  2.63469954e-
01  1.38083522e-01
 -5.15121315e-02 -1.31779269e-01  5.93215550e-
04 -1.50770169e-02
 -2.11866755e-02 -1.31314650e-02  6.68837224e-
02  6.27486301e-02
  5.25005262e-02 -2.61701636e-02 -1.23109025e-
02 -5.59755788e-02
 -1.53817325e-01 -5.57476829e-02  8.52850593e-
02 -1.49248075e-01
```

```
  4.19563587e-02  8.75441596e-02]
```

In [73]:

```
import pickle
f=open('11svd_matrix_essay.pckl','wb')
pickle.dump([essay_vectors],f)
f.close()
```

In [98]:

```
import pickle as pickle
#with open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai course/DonorsChoose_2018/cat_num.pckl', 'rb') as f:
f=open('C:/Users/pramod reddy chandi/Desktop/pram/applied ai course/DonorsChoose_2018/11svd_matrix_essay.pckl','rb')
essay_vectors=pickle.load(f)
f.close()
```

In [100]:

```
len(essay_vectors)
```

Out[100]:

```
1
```

In [101]:

```
data1=[]
for x in essay_vectors:
    data1.extend(x)
```

In [102]:

```
import pandas as pd
df1 = pd.DataFrame(data1)
```

In [103]:

```
df1.shape
```

Out[103]:

```
(109248, 250)
```

In [95]:

```
i=0
list_of_sentance_train=[]
for sentance in process_columns['preprocessed_title']:
    list_of_sentance_train.append(sentance.split())
```

In [96]:

```
# average Word2Vec of essays
# compute average word2vec for each review.
title_vectors= []; # the avg-w2v for each sentence/review is stored in this list
for sent in tqdm(list_of_sentance_train): # for each review/sentence
    sent_vec = np.zeros(250) # as word vectors are of zero length 50, you might need to change this to 300 if you use google's w2v
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words:
            i=top_features[top_features['Top Words'] == word].index.tolist()
            vec=df.iloc[:,i]
            sent_vec += vec.values.reshape(250)
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    title_vectors.append(sent_vec)
title_vectors= np.array(title_vectors)
print(title_vectors.shape)
print(title_vectors[0])
```

```
100%|██████████| 109248/109248 [3:03:20<00:00,
 10.02it/s]
```

```
(109248, 250)
[ 9.74172235e+01  5.32331148e-01 -1.45490748e+
00  4.20405267e+00
  6.16255411e+00 -8.37733206e+00  3.99740634e+
00 -2.06031163e+00
  1.16060315e+00 -1.84187262e+00 -2.79692835e+
00  7.33901305e+00
  3.35574965e+00 -3.60079680e+00 -1.25726436e+
00  2.08380949e+00
 -4.24189828e-01  7.47716708e-01  2.74757554e+
00  2.45053494e-01
  2.63811095e+00 -1.82878551e-01  1.75973527e-
01 -6.68674893e-01
 -1.40934098e-01 -1.14356107e+00 -5.51414338e-
01  3.51324457e-01
 -3.38385196e-01  5.94269042e-01 -1.25762678e+
00  4.31334755e-01
  6.97566512e-02 -5.58107316e-02 -4.56440493e-
02  1.27821158e-01
  4.03623443e-01  7.09484702e-02  9.03657140e-
02 -5.40737442e-01
  7.39934629e-02 -4.90720794e-01  1.25234494e-
01  9.95381638e-02
 -1.43709152e-02 -4.86235260e-01 -5.03471335e-
01  8.79164187e-02
 -1.57751181e-01 -2.23817452e-01  1.32316487e-
02  3.95347873e-01
 -1.13700848e-01 -1.80364711e-01 -2.50526365e-
01 -1.93476265e-02
 -4.40756673e-03  2.33762182e-01  5.16696072e-
02  6.20313511e-03
 -1.02182785e-01 -1.05914084e-01  2.12334914e-
02  7.99744904e-02
```

```
 -8.77202085e-02 -6.40117852e-02  9.06807786e-
02 -2.30628470e-01
  2.05672383e-02 -1.59556699e-01 -1.99729209e-
02  9.43785099e-03
 -3.72660528e-02  1.35116335e-01  4.27319772e-
02 -1.20334040e-01
  5.95472691e-02  2.20117951e-02  2.04706605e-
02  4.63296395e-02
 -1.38854143e-01  5.35979070e-02  1.44901355e-
01 -8.41433896e-03
  2.84223386e-02  1.30284915e-02  7.75789160e-
02 -9.50100512e-02
 -1.02364414e-02 -2.20238583e-02 -1.29729944e-
01 -7.85970266e-02
  2.23137858e-02  3.16134853e-02  1.09472691e-
01 -3.25119178e-02
  8.91847462e-02 -1.66937391e-03  1.17462969e-
01 -1.63279928e-02
 -9.12034147e-03  3.19031476e-02  5.06866437e-
02  1.19037687e-01
 -1.01629457e-01 -4.76538444e-02  1.15584643e-
01  3.72782990e-02
  9.93942196e-02 -1.26592362e-01  5.99156418e-
02  3.82421818e-02
  1.88629921e-03  7.98315286e-02  4.69970617e-
02 -2.36469086e-02
 -6.05666122e-03 -3.12573075e-02  1.36127409e-
02  8.89954518e-03
 -8.87936057e-03  2.97593632e-02  2.30415602e-
03 -1.64664082e-02
 -1.41853039e-02 -1.70025947e-02  1.67923405e-
02 -5.02304509e-02
 -1.52588322e-02  4.18213157e-02  1.70617959e-
02  3.24452933e-02
 -2.76637331e-03 -4.66516610e-03  2.58157715e-
02 -1.85133001e-02
  8.02951402e-03  3.67205151e-02 -1.25735969e-
```

```
02  1.51425465e-02
 -8.84310068e-03 -5.75966208e-02 -3.94414840e-
02  1.73850522e-02
  3.47626847e-02 -4.36782455e-02 -2.99004119e-
02  8.16711022e-03
  7.53071877e-03  2.26078036e-02  3.95073374e-
02 -6.29668382e-03
  1.95231273e-02  6.56378030e-02 -7.62892257e-
03  4.00894590e-03
  1.79773054e-02  8.59271600e-03 -2.48297716e-
04 -1.28216753e-02
 -3.63515043e-02 -2.78366605e-02  2.53982534e-
02 -1.04447790e-02
  4.79864661e-03 -2.56373663e-02 -9.93178436e-
03  6.06559209e-03
  2.92086516e-03 -2.57342396e-02  1.38603179e-
02  5.03692589e-02
  1.51982465e-02 -1.11031382e-02 -2.89399452e-
02 -2.10226653e-02
 -3.30327074e-02  9.84953043e-03  2.61622723e-
02  1.14755868e-02
  3.18262703e-02 -5.31670004e-03  1.96762255e-
02  7.01390224e-03
 -2.91848424e-02 -1.34614441e-02  8.10951222e-
03 -3.67385249e-03
  1.18925296e-02 -5.55098747e-03  3.01868284e-
02  4.39859199e-03
 -1.01961009e-02 -2.94119336e-02 -2.08991598e-
02  5.40152819e-03
  1.99135800e-03  4.71871118e-03  8.20182443e-
03  4.65626144e-03
 -2.02069580e-02 -1.47950401e-02  1.81858488e-
02 -4.28720416e-03
  2.11803128e-02 -1.24762726e-02  7.61514820e-
04 -2.15861974e-03
 -8.91106636e-03  2.03028026e-03 -2.19553887e-
02  1.01886794e-02
```

```
 -6.08005000e-03 -1.01465379e-02 -1.71367800e-
02  3.85731540e-02
 -1.00304066e-02  2.68461787e-02 -7.84338953e-
03 -1.11167156e-03
 -6.18180199e-03 -1.10562913e-02 -7.64228844e-
03  1.14962338e-02
  9.77025470e-03  1.74635561e-02  1.68733971e-
02 -6.87842228e-03
  2.78166028e-02  1.94692237e-03  1.17747816e-
02 -7.04978770e-03
  5.88151319e-03 -2.09360884e-02 -1.49250658e-
02 -1.69009071e-02
  3.54120268e-05 -8.87763704e-03  2.83053684e-
03 -1.92237177e-02
  8.54248915e-03 -3.10142176e-02 -1.79351442e-
03 -1.26030943e-02
 -7.34064349e-03 -3.53138330e-03  1.06020725e-
02  5.21820240e-03
  1.69343225e-02  1.14764995e-02]
```

In [97]:

```
import pickle
f=open('11svd_matrix_title.pckl','wb')
pickle.dump([title_vectors],f)
f.close()
```

In [104]:

```
title_vectors.shape
```

Out[104]:

```
(109248, 250)
```

In [105]:

```
type(title_vectors)
```

Out[105]:

```
numpy.ndarray
```

In [109]:

```
title_vectors.shape
```

Out[109]:

```
(109248, 250)
```

In [106]:

```
type(df1)
```

Out[106]:

```
pandas.core.frame.DataFrame
```

In [107]:

```
df1.shape
```

Out[107]:

```
(109248, 250)
```

In [115]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler(with_mean=True).fit(df1.values)
df_essay= scalar.transform(df1.values)
```

In [110]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler(with_mean=True).fit(title_vectors)
df_title= scalar.transform(title_vectors)
```

In [121]:

```
from scipy.sparse import hstack
```

```
step4_vector_standard=hstack((cat_num,df_essay,df_title))
```

In [122]:

```
step4_vector_standard.shape
```

Out[122]:

```
(109248, 608)
```

In [123]:

```
type(step4_vector_standard)
```

Out[123]:

```
scipy.sparse.coo.coo_matrix
```

In [124]:

```
step4_vector_standard
```

Out[124]:

```
<109248x608 sparse matrix of type '<class 'num
py.float64'>'
        with 56224006 stored elements in COOrd
inate format>
```

In [125]:

```
import scipy.sparse
scipy.sparse.save_npz('step4_vector_standard_values.npz', ste
p4_vector_standard)
```

In [1]:

```
import scipy.sparse
sparse_matrix = scipy.sparse.load_npz('step4_vector_standard_
values.npz')
```

In [2]:

```
sparse_matrix.shape
```

Out[2]:

```
(109248, 608)
```

In [3]:

```
sparse_matrix
```

Out[3]:

```
<109248x608 sparse matrix of type '<class 'num
py.float64'>'
        with 56224006 stored elements in COOrd
inate format>
```

In [4]:

```
type(sparse_matrix)
```

Out[4]:

```
scipy.sparse.coo.coo_matrix
```

Assignment 11: TruncatedSVD

- step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title and then find the top 2k words) based on their `idf_` values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref)
- step 3 Use TruncatedSVD on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method

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

- step 4 Concatenate these truncatedSVD matrix, with the matrix with features

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

In [5]:

```
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():
```

```python
    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=1)

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

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

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

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

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

In [13]:

```
#splitting the data into train and test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(sparse_ma
trix, Y, test_size=0.2, random_state=42)
```

In [14]:

```
classifier = XGBoostClassifier(eval_metric = 'auc', num_class
 = 2, nthread = 4,
                               silent=False,
                      scale_pos_weight=1,
                      learning_rate=0.01,
                      colsample_bytree = 0.4,
                      subsample = 0.8,
                      n_estimators=1000,
                      reg_alpha = 0.3,
                      max_depth=3,
                      gamma=10)
```

In [133]:

```
classifier.fit(X_train,y_train)
```

In [134]:

```
classifier.score(X_train,y_train)
```

Out[134]:

```
0.6206566764934691
```

In [135]:

```
classifier.score(X_test,y_test)
```

Out[135]:

```
0.6055360433978995
```

we get accuracy of over 60 % both on train and test data respectively.let us tune the hyperparameters and check the results.

In [15]:

```
#tuning max depth hyperparameter
import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.metrics import roc_auc_score
import math

train_auc = []
test_auc = []

K = [3, 4, 5, 6, 8, 10, 12, 15]
for i in K:
    classifier = XGBoostClassifier(eval_metric = 'auc', num_c
lass = 2, nthread = 4,
                               silent=False,
                      scale_pos_weight=1,
                      learning_rate=0.01,
                      colsample_bytree = 0.4,
                      subsample = 0.8,
                      n_estimators=1000,
                      reg_alpha = 0.3,
                      max_depth=i,
                      gamma=10)

    classifier.fit(X_train, y_train)
    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(classifier.score(X_train,y_train))
    test_auc.append(classifier.score(X_test,y_test))
```

In [16]:

```
train_auc
```

Out[16]:

```
[0.6206566764934691,
 0.6293961665034373,
 0.6337327667882224,
 0.6354154849473798,
 0.6361381528895538,
 0.6368607578614314,
 0.6375447402051494,
 0.6375447402051494]
```

In [17]:

```
test_auc
```

Out[17]:

```
[0.6055360433978995,
 0.6112156382008535,
 0.6127612378873393,
 0.614076741833752,
 0.6142075516087621,
 0.6143287341542283,
 0.6143145079822696,
 0.6143145079822696]
```

In [18]:

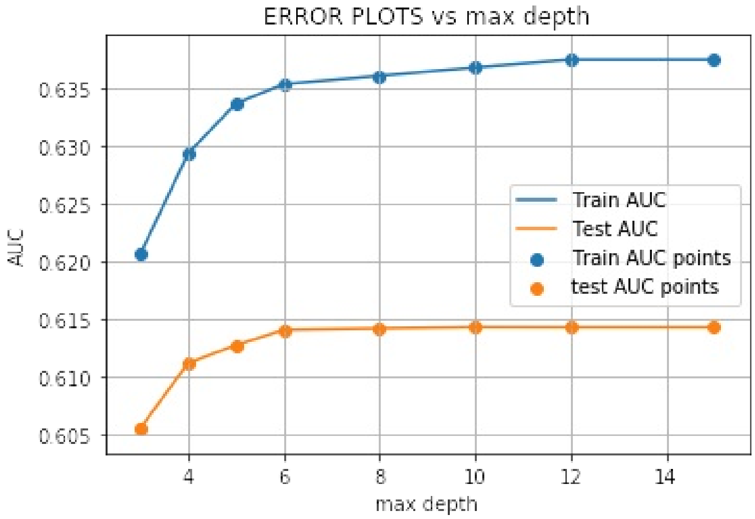
```
K
```

Out[18]:

```
[3, 4, 5, 6, 8, 10, 12, 15]
```

In [19]:

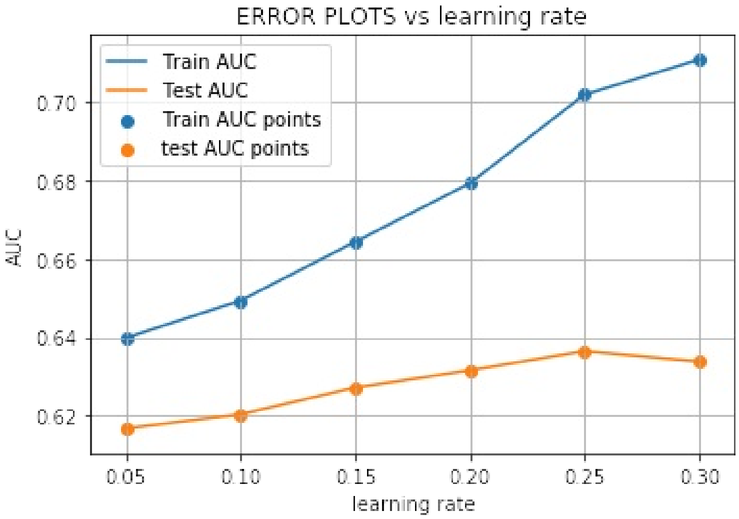
```
plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, test_auc, label='Test AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, test_auc, label='test AUC points')

plt.legend()
plt.xlabel("max depth")
plt.ylabel("AUC")
plt.title("ERROR PLOTS vs max depth")
plt.grid()
plt.show()
```



we get for max depth of 12 we get good performance as it is giving max yield.

In [20]:

```
#tuning learning rate hyperparameter

import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.metrics import roc_auc_score
import math

train_auc1 = []
test_auc1= []

l = [0.05, 0.10, 0.15, 0.20, 0.25, 0.30 ]
for i in l:
    classifier = XGBoostClassifier(eval_metric = 'auc', num_c
lass = 2, nthread = 4,
                               silent=False,
                      scale_pos_weight=1,
                      learning_rate= i,
                      colsample_bytree = 0.4,
                      subsample = 0.8,
                      n_estimators=1000,
                      reg_alpha = 0.3,
                      max_depth=12,
                      gamma=10)

    classifier.fit(X_train, y_train)
    # roc_auc_score(y_true, y_score) the 2nd parameter +should be probability estimates of the positive class
    # not the predicted outputs
    train_auc1.append(classifier.score(X_train,y_train))
    test_auc1.append(classifier.score(X_test,y_test))

plt.plot(l, train_auc1, label='Train AUC')
plt.plot(l, test_auc1, label='Test AUC')

plt.scatter(l, train_auc1, label='Train AUC points')
plt.scatter(l, test_auc1, label='test AUC points')

plt.legend()
plt.xlabel("learning rate")
plt.ylabel("AUC")
plt.title("ERROR PLOTS vs learning rate")
plt.grid()
plt.show()
```

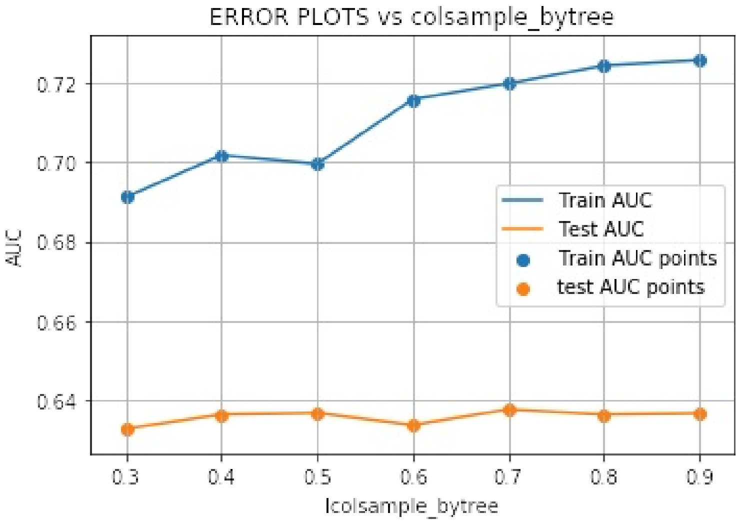


we could see that the best hyperparameter is 0.25

In [21]:

```
#tuning colsample_bytree yperparameter

import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.metrics import roc_auc_score
import math

train_auc2= []
test_auc2 =[]

l = [0.3,0.4, 0.5, 0.6, 0.7, 0.8, .9]
for i in l:
    classifier = XGBoostClassifier(eval_metric = 'auc', num_c
lass = 2, nthread = 4,
                               silent=False,
                      scale_pos_weight=1,
                      learning_rate= 0.25,
                      colsample_bytree = i,
                      subsample = 0.8,
                      n_estimators=1000,
                      reg_alpha = 0.3,
                      max_depth=12,
                      gamma=10)

    classifier.fit(X_train, y_train)


    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
```

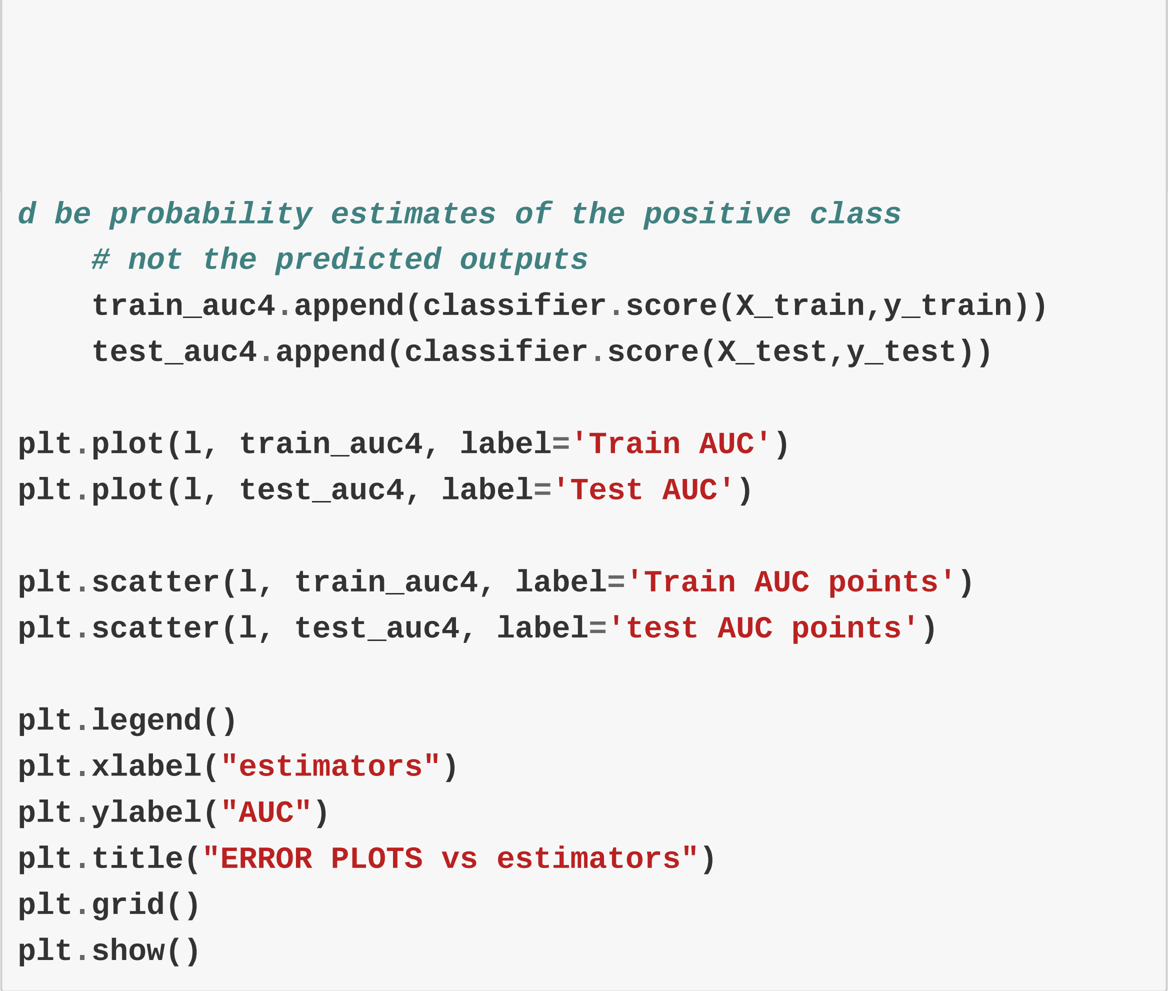
```
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc2.append(classifier.score(X_train,y_train))
    test_auc2.append(classifier.score(X_test,y_test))

plt.plot(l, train_auc2, label='Train AUC')
plt.plot(l, test_auc2, label='Test AUC')

plt.scatter(l, train_auc2, label='Train AUC points')
plt.scatter(l, test_auc2, label='test AUC points')

plt.legend()
plt.xlabel("lcolsample_bytree")
plt.ylabel("AUC")
plt.title("ERROR PLOTS vs colsample_bytree")
plt.grid()
plt.show()
```



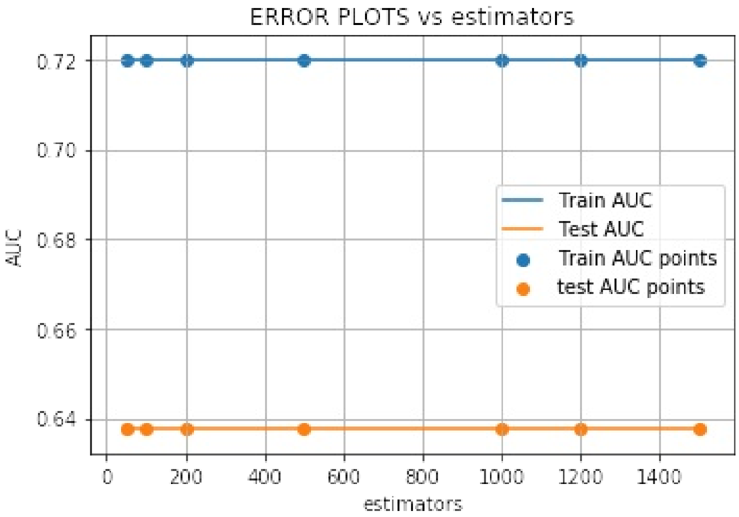
we can see that the best hyperparameter is 0.7

In [22]:

```
#tuning colsample_bytree yperparameter

import matplotlib.pyplot as plt
import xgboost as xgb
from sklearn.metrics import roc_auc_score
import math

train_auc4= []
test_auc4 =[]

l = [50,100,200,500,1000,1200,1500]
for i in l:
    classifier = XGBoostClassifier(eval_metric = 'auc', num_c
lass = 2, nthread = 4,
                               silent=False,
                      scale_pos_weight=1,
                      learning_rate= 0.25,
                      colsample_bytree = 0.7,
                      subsample = 0.8,
                      n_estimators= i,
                      reg_alpha = 0.3,
                      max_depth=12,
                      gamma=10)

    classifier.fit(X_train, y_train)


    # roc_auc_score(y_true, y_score) the 2nd parameter +shoul
```



```
d be probability estimates of the positive class
    # not the predicted outputs
    train_auc4.append(classifier.score(X_train,y_train))
    test_auc4.append(classifier.score(X_test,y_test))

plt.plot(l, train_auc4, label='Train AUC')
plt.plot(l, test_auc4, label='Test AUC')

plt.scatter(l, train_auc4, label='Train AUC points')
plt.scatter(l, test_auc4, label='test AUC points')

plt.legend()
plt.xlabel("estimators")
plt.ylabel("AUC")
plt.title("ERROR PLOTS vs estimators")
plt.grid()
plt.show()
```



we could see that there is no much change in performance with change in estimators

In [23]:

```
#final model
classifier = XGBoostClassifier(eval_metric = 'auc', num_class
 = 2, nthread = 4,
                               silent=False,
                      scale_pos_weight=1,
                      learning_rate= 0.25,
                      colsample_bytree = 0.7,
                      subsample = 0.8,
                      n_estimators= 100,
                      reg_alpha = 0.3,
                      max_depth=12,
                      gamma=10)
classifier.fit(X_train, y_train)
train_auc_final1=classifier.score(X_train,y_train)
test_auc_final1=classifier.score(X_test,y_test)
```

In [24]:

```
train_auc_final1
```

Out[24]:

```
0.719863001924743
```

In [25]:

```
test_auc_final1
```

Out[25]:

```
0.6376774492022866
```

3. Conclusion

Please write down few lines about what you observed from this assignment.

we could see a significant improvement in the model with just using the top 2k features of essay and title and we only considered 250 features of it.Thus we can conclude that only few features contribute lot to the final model .

We can ignore less important features .

we can also find that xgboost classifier is fast compared to other models.