prabhudayala@gmail.com_11

June 15, 2019

1 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

How to scale current manual processes and resources to screen 500,000 projects so that they ca
<li>How to increase the consistency of project vetting across different volunteers to improve
<li>How to focus volunteer time on the applications that need the most assistance</li>
</ul>
```

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.

1.1 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`

Feature	Description
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.

1.1.1 Notes on the Essay Data

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

project_essay_1: "Introduce us to your classroom"

project_essay_2: "Tell us more about your students"

project_essay_3: "Describe how your students will use the materials you're requesting"

project_essay_3: "Close by sharing why your project will make a difference"

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

project_essay_1: "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

project_essay_2: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

```
In [1]: %matplotlib inline
        import warnings
```

```
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.2 1.1 Reading Data

```
In [2]: project_data = pd.read_csv('train_data.csv')
        resource_data = pd.read_csv('resources.csv')
```

```
In [3]: print("Number of data points in train data", project_data.shape)
        print('-'*50)
        print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
```

```
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:
```
        id                                        description  quantity  \
0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
1  p069063        Bouncy Bands for Desks (Blue support pipes)         3

    price
0  149.00
1   14.95
```

In [5]:
```
# join two dataframes in python:
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_i
price_data.head(2)
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [6]:
```
project_data.columns
```

Out[6]:
```
Index(['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',
       'price', 'quantity'],
      dtype='object')
```

1.3 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/4

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-str
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyt
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", "Warmt
        if 'The' in j.split(): # this will split each of the catogory based on space "
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing sp
        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]))
```

1.4 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/4

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

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", "Warmt
        if 'The' in j.split(): # this will split each of the catogory based on space "
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing sp
        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/22898595/408403
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]))
```

1.5 1.3 Text preprocessing

In [9]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [10]:
```
project_data.head(2)
```

Out[10]:
```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.

  school_state project_submitted_datetime project_grade_category  \
0           IN        2016-12-05 13:43:57          Grades PreK-2
1           FL        2016-10-25 09:22:10             Grades 6-8

                                    project_title  \
0  Educational Support for English Learners at Home
1             Wanted: Projector for Hungry Learners

                                  project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...

                                  project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...             NaN
1  The projector we need for our school is very c...             NaN

  project_essay_4                         project_resource_summary  \
0             NaN  My students need opportunities to practice beg...
1             NaN  My students need a projector to help with view...

   teacher_number_of_previously_posted_projects  project_is_approved  price  \
0                                             0                    0  154.6
1                                             7                    1  299.0

   quantity              clean_categories           clean_subcategories  \
0        23             Literacy_Language                  ESL Literacy
1         1  History_Civics Health_Sports  Civics_Government TeamSports
```

```
                                               essay
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...
```

In [11]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [12]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, a
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogn
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The g
==================================================
```

In [13]:
```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

```
In [14]: sent = decontracted(project_data['essay'].values[20000])
         print(sent)
         print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogr
==================================================
```

```
In [15]: # \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-p
         sent = sent.replace('\\r', ' ')
         sent = sent.replace('\\"', ' ')
         sent = sent.replace('\\n', ' ')
         print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogr
```

```
In [16]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
         sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
         print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cogni
```

```
In [17]: # https://gist.github.com/sebleier/554280
         # we are removing the words from the stop words list: 'no', 'nor', 'not'
         stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'
                     "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him'
                     'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
                     'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "
                     'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'l
                     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
                     'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
                     'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'c
                     'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'ar
                     'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
                     's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'r
                     've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't"
                     "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mig
                     "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
                     'won', "won't", 'wouldn', "wouldn't"]
```

```
In [18]: # Combining all the above stundents
         from tqdm import tqdm
         preprocessed_essays = []
         # tqdm is for printing the status bar
         for sentance in tqdm(project_data['essay'].values):
             sent = decontracted(sentance)
             sent = sent.replace('\\r', ' ')
```

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

```
100%|| 109248/109248 [00:44<00:00, 2436.28it/s]
```

```
In [19]: # after preprocesing
         preprocessed_essays[20000]
```

```
Out[19]: 'my kindergarten students varied disabilities ranging speech language delays cognitive
```

```
In [20]: project_data["essay"]=preprocessed_essays
```

1.4 Preprocessing of project_title

```
In [21]: from tqdm import tqdm
         preprocessed_project_title = []
         # tqdm is for printing the status bar
         for sentance in tqdm(project_data['project_title'].values):
             sent = decontracted(sentance)
             sent = sent.replace('\\r', ' ')
             sent = sent.replace('\\"', ' ')
             sent = sent.replace('\\n', ' ')
             sent = 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_project_title.append(sent.lower().strip())
```

```
100%|| 109248/109248 [00:01<00:00, 55350.51it/s]
```

```
In [22]: print(project_data['project_title'].values[20000])
         project_data['project_title']=preprocessed_project_title
         print(project_data['project_title'].values[20000])
```

```
We Need To Move It While We Input It!
need move input
```

```
In [23]: from nltk.sentiment import SentimentIntensityAnalyzer as SID
         #nltk.download('vader_lexicon')
         new_df_as_dictinary=[]
         sid=SID()
         for i in tqdm(project_data.essay.values):
             new_df_as_dictinary.append(sid.polarity_scores(i))
```

```
100%|| 109248/109248 [02:32<00:00, 714.80it/s]
```

```
In [24]: print(project_data.columns)
         print(project_data.shape)
         sentiment_score=pd.DataFrame(new_df_as_dictinary)
         print(sentiment_score.columns)
         print(sentiment_score.shape)
```

```
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',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'essay'],
      dtype='object')
(109248, 20)
Index(['compound', 'neg', 'neu', 'pos'], dtype='object')
(109248, 4)
```

```
In [25]: sentiment_score=pd.DataFrame(new_df_as_dictinary)
         project_data=pd.concat((project_data,sentiment_score),axis=1,ignore_index=True)
         print(project_data.shape)
```

```
(109248, 24)
```

```
In [26]: project_data.columns=['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_sta
                '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',
                'price', 'quantity', 'clean_categories', 'clean_subcategories',
                'essay','compound', 'neg', 'neu', 'pos']
```

```
In [27]: # for i,j in enumerate(preprocessed_project_title)
         project_data['combined_essay_title']=project_data['project_title']+" "+project_data['
```

```
In [28]: print(project_data['essay'].values[20000])
         print(project_data['project_title'].values[20000])
         print(project_data['combined_essay_title'].values[20000])
```

```
my kindergarten students varied disabilities ranging speech language delays cognitive delays g
need move input
need move input my kindergarten students varied disabilities ranging speech language delays co
```

2 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 [31]:
```python
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_bo

    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,verbose=10,n_jobs=4)
X = np.array([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])
```

```python
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]))
```

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits


[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   5 tasks      | elapsed:    2.1s
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:    3.3s
[Parallel(n_jobs=4)]: Done  17 tasks      | elapsed:    6.7s
[Parallel(n_jobs=4)]: Done  24 tasks      | elapsed:    7.9s
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:   12.2s
[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:   13.6s
[Parallel(n_jobs=4)]: Done  53 tasks      | elapsed:   18.8s
```

```
    ---------------------------------------------------------------------------

    KeyboardInterrupt                         Traceback (most recent call last)

    <ipython-input-31-ea2b716278a7> in <module>()
     63 X = np.array([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])
     64 Y = np.array([0, 1, 0, 1, 0, 1])
---> 65 clf.fit(X, Y)
     66
     67 # print(clf.grid_scores_)


    ~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in fit(self, X, y, gro
    720                 return results_container[0]
    721
--> 722             self._run_search(evaluate_candidates)
    723
    724         results = results_container[0]


    ~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in _run_search(self, e
   1189     def _run_search(self, evaluate_candidates):
   1190         """Search all candidates in param_grid"""
-> 1191         evaluate_candidates(ParameterGrid(self.param_grid))
   1192
```

```
   1193


    ~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in evaluate_candidate
    709                                 for parameters, (train, test)
    710                                 in product(candidate_params,
--> 711                                            cv.split(X, y, groups)))
    712
    713                 all_candidate_params.extend(candidate_params)


    ~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in __call__(self, i
    928
    929             with self._backend.retrieval_context():
--> 930                 self.retrieve()
    931             # Make sure that we get a last message telling us we are done
    932             elapsed_time = time.time() - self._start_time


    ~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in retrieve(self)
    831             try:
    832                 if getattr(self._backend, 'supports_timeout', False):
--> 833                     self._output.extend(job.get(timeout=self.timeout))
    834                 else:
    835                     self._output.extend(job.get())


    ~\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py in wrap_f
    519         AsyncResults.get from multiprocessing."""
    520         try:
--> 521             return future.result(timeout=timeout)
    522         except LokyTimeoutError:
    523             raise TimeoutError()


    ~\Anaconda3\lib\concurrent\futures\_base.py in result(self, timeout)
    425                 return self.__get_result()
    426
--> 427             self._condition.wait(timeout)
    428
    429             if self._state in [CANCELLED, CANCELLED_AND_NOTIFIED]:


    ~\Anaconda3\lib\threading.py in wait(self, timeout)
    293         try:    # restore state no matter what (e.g., KeyboardInterrupt)
    294             if timeout is None:
--> 295                 waiter.acquire()
    296                 gotit = True
```

```
297             else:


KeyboardInterrupt:
```

2. TruncatedSVD

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

In [29]:
```
project_data.columns
```

Out[29]:
```
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',
       'price', 'quantity', 'clean_categories', 'clean_subcategories', 'essay',
       'compound', 'neg', 'neu', 'pos', 'combined_essay_title'],
      dtype='object')
```

In [66]:
```
sampling=False
undersampling=True
if (not sampling):
    print("Total data ",project_data.shape)

else:
    if(sampling and undersampling):
        print("Total data ",project_data.shape)
        project_data_negative=project_data[project_data.project_is_approved==0]
        project_data_positive=project_data[project_data.project_is_approved==1]
        project_data_positive=project_data_positive.sample(n=project_data_negative.sha
        print("Positive points: ",project_data_positive.shape[0])
        print("Negaitive points: ",project_data_negative.shape[0])
        project_data=pd.concat([project_data_positive,project_data_negative])
    else:
        print("Total data ",project_data.shape)
        project_data_negative=project_data[project_data.project_is_approved==0]
        project_data_positive=project_data[project_data.project_is_approved==1]
        project_data_negative=project_data_negative.sample(n=project_data_positive.sha
        print("Positive points: ",project_data_positive.shape[0])
        print("Negaitive points: ",project_data_negative.shape[0])
        project_data=pd.concat([project_data_positive,project_data_negative])

data_point_size=10000
project_data=project_data.sample(n=data_point_size,random_state=42,replace=True)
print("positive and negative counts")
print(project_data.project_is_approved.value_counts())
project_data_Y=project_data.project_is_approved
```

```
#project_data_X=project_data.drop(columns=['project_is_approved'])
project_data_X=project_data
print("After sampling: ",project_data_X.shape)
```

```
Total data  (109248, 25)
positive and negative counts
1    8564
0    1436
Name: project_is_approved, dtype: int64
After sampling:  (10000, 25)
```

In [67]:
```
from sklearn.model_selection import train_test_split
project_data_X_train,project_data_X_test,project_data_Y_train,project_data_Y_test=tra
```

2.1 Selecting top 2000 words from essay and `project_title`

In [68]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_combined_essay_title_tfidf = TfidfVectorizer()
vectorizer_combined_essay_title_tfidf.fit(project_data_X_train.combined_essay_title.va
```

Out[68]:
```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), norm='l2', preprocessor=None, smooth_idf=True,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=True,
        vocabulary=None)
```

In [69]:
```
#use idf to choose top 2000 features
tfidf_score = vectorizer_combined_essay_title_tfidf.idf_
tfidf_score_argsort=np.argsort(vectorizer_combined_essay_title_tfidf.idf_)[::-1]
tfidf_score_argsort = tfidf_score_argsort[:2000]

all_words_tfidf=vectorizer_combined_essay_title_tfidf.get_feature_names()
all_words_tfidf=np.array(all_words_tfidf)
top2000_words= list(all_words_tfidf[tfidf_score_argsort])
print(len(top2000_words))
```

```
2000
```

2.2 Computing Co-occurance matrix

In [74]:
```
def co_occurence_matrix(win,vocab,corpus,coo_matrix):
    window=win
    a=vocab
    for q,word in enumerate(vocab):
        print("Word number ",q,word)
```

```
            for i in corpus.values:
                if word in i:
                    arr=[g for g in i.split(' ')]
                    for j,d in enumerate(arr):
                        arrr=[]
                        for i in range(max(0,j-window),min(j+window,len(arr)-1)):
                            arrr.append(arr[i])
                            for f,wd in enumerate(arrr):
                                if wd in vocab:
                                    if wd!=word:
                                        index=vocab.index(wd)
                                        coo_matrix[q,index]+=1
        return coo_matrix
```

In [75]:
```
cooc_train=np.zeros((2000,2000))
print(cooc_train.shape)
```

```
(2000, 2000)
```

In [76]:
```
cooc_train=co_occurence_matrix(5,top2000_words,project_data_X_train.combined_essay_ti
```

```
Word number  0 zz
Word number  1 fad
Word number  2 reinvigorates
Word number  3 reiterate
Word number  4 reject
Word number  5 rejoice
Word number  6 rejoicing
Word number  7 rejuvenate
Word number  8 rekenerks
Word number  9 fadeless
Word number  10 rekindle
Word number  11 rekindled
Word number  12 baseplates
Word number  13 baselines
Word number  14 chickering
Word number  15 dashing
Word number  16 childish
Word number  17 chicopee
Word number  18 chid
Word number  19 factoring
Word number  20 bartholomew
Word number  21 barth
Word number  22 dater
Word number  23 barry
Word number  24 relaxes
Word number  25 childeren
Word number  26 loser
```

```
Word number  27 barrio
Word number  28 relayed
Word number  29 relays
Word number  30 reinvesting
Word number  31 chickadees
Word number  32 graaff
Word number  33 reinvent
Word number  34 batches
Word number  35 chex
Word number  36 fafsa
Word number  37 chez
Word number  38 lovelearning
Word number  39 regulators
Word number  40 lovelace
Word number  41 chezy
Word number  42 chibitronics
Word number  43 chichagof
Word number  44 bassoon
Word number  45 reheating
Word number  46 reignite
Word number  47 reigns
Word number  48 reindeer
Word number  49 bassick
Word number  50 basking
Word number  51 louv
Word number  52 loungin
Word number  53 basiswe
Word number  54 basin
Word number  55 reinforment
Word number  56 reinisch
Word number  57 reinstated
Word number  58 reintroduce
Word number  59 reintroduced
Word number  60 reintroduction
Word number  61 barren
Word number  62 relegated
Word number  63 faigenbaum
Word number  64 chills
Word number  65 bard
Word number  66 barcode
Word number  67 barber
Word number  68 grade5
Word number  69 facetime
Word number  70 chiller
Word number  71 dawn
Word number  72 remediated
Word number  73 lookin
Word number  74 barb
```

Word number 75 remediations
Word number 76 remedies
Word number 77 remedy
Word number 78 baptism
Word number 79 relented
Word number 80 longstanding
Word number 81 banquet
Word number 82 facelifts
Word number 83 banners
Word number 84 banner
Word number 85 longmeadow
Word number 86 reminds
Word number 87 longitude
Word number 88 remission
Word number 89 remnants
Word number 90 remodel
Word number 91 remodeled
Word number 92 banneker
Word number 93 bargain
Word number 94 bargains
Word number 95 reluctantly
Word number 96 baritone
Word number 97 lorain
Word number 98 facilitators
Word number 99 looting
Word number 100 relevants
Word number 101 relevent
Word number 102 barnyard
Word number 103 loot
Word number 104 looses
Word number 105 loosen
Word number 106 looseleaf
Word number 107 childreni
Word number 108 barnhill
Word number 109 loopy
Word number 110 childrennannan
Word number 111 loophole
Word number 112 barley
Word number 113 baritones
Word number 114 childrens
Word number 115 relining
Word number 116 relinquish
Word number 117 davenport
Word number 118 relive
Word number 119 reliving
Word number 120 looming
Word number 121 faciliate
Word number 122 relocating

```
Word number  123 reluctance
Word number  124 batelle
Word number  125 chews
Word number  126 axles
Word number  127 reenact
Word number  128 danish
Word number  129 beacons
Word number  130 familiarizing
Word number  131 reductions
Word number  132 reductive
Word number  133 redundancy
Word number  134 luggage
Word number  135 redwood
Word number  136 lug
Word number  137 beacon
Word number  138 familar
Word number  139 reels
Word number  140 reemphasizing
Word number  141 luczynski
Word number  142 dao
Word number  143 reenergize
Word number  144 reenforce
Word number  145 reengaged
Word number  146 reevaluate
Word number  147 reevaluating
Word number  148 reexamine
Word number  149 beachy
Word number  150 beaches
Word number  151 referenced
Word number  152 famed
Word number  153 referencing
Word number  154 lucy
Word number  155 chet
Word number  156 beaded
Word number  157 beaders
Word number  158 reds
Word number  159 redox
Word number  160 lunchbox
Word number  161 fanning
Word number  162 fannin
Word number  163 recreationally
Word number  164 recruit
Word number  165 recruiters
Word number  166 recruitment
Word number  167 chestnut
Word number  168 lumos
Word number  169 rectify
Word number  170 fangled
```

Word number 171 fanciful
Word number 172 fanatics
Word number 173 lummi
Word number 174 beakers
Word number 175 beaker
Word number 176 lumi
Word number 177 redeployment
Word number 178 lumberton
Word number 179 redesignated
Word number 180 redesigned
Word number 181 lulzbot
Word number 182 lukow
Word number 183 familynannan
Word number 184 daniel
Word number 185 redirections
Word number 186 redlin
Word number 187 falters
Word number 188 refilling
Word number 189 das
Word number 190 regimens
Word number 191 refused
Word number 192 refusing
Word number 193 refute
Word number 194 fairbairn
Word number 195 regal
Word number 196 darkening
Word number 197 baths
Word number 198 gown
Word number 199 regency
Word number 200 regent
Word number 201 lowell
Word number 202 gowns
Word number 203 regime
Word number 204 regiment
Word number 205 dapping
Word number 206 regimented
Word number 207 regimes
Word number 208 reginald
Word number 209 bathmats
Word number 210 lovingly
Word number 211 darkroom
Word number 212 registered
Word number 213 registering
Word number 214 lovies
Word number 215 registery
Word number 216 regret
Word number 217 regreted
Word number 218 darrow

```
Word number  219 fairfax
Word number  220 refurbishing
Word number  221 gowan
Word number  222 fairfield
Word number  223 bdeir
Word number  224 refinement
Word number  225 refines
Word number  226 fallacies
Word number  227 bball
Word number  228 lsu
Word number  229 baytown
Word number  230 bayside
Word number  231 dared
Word number  232 ls1
Word number  233 falcon
Word number  234 falafel
Word number  235 reflexes
Word number  236 darell
Word number  237 lpms
Word number  238 lows
Word number  239 reform
Word number  240 reformed
Word number  241 refractometers
Word number  242 refrain
Word number  243 refraining
Word number  244 governs
Word number  245 lowes
Word number  246 refresher
Word number  247 fairmount
Word number  248 daring
Word number  249 fairies
Word number  250 facelift
Word number  251 longevity
Word number  252 removal
Word number  253 littlebit
Word number  254 livable
Word number  255 litwin
Word number  256 littleton
Word number  257 backsack
Word number  258 deadly
Word number  259 cots
Word number  260 backrounds
Word number  261 backpacking
Word number  262 backpacker
Word number  263 responsibiltiy
Word number  264 responsibitlity
Word number  265 backlit
Word number  266 responsiblitly
```

Word number 267 backings
Word number 268 restrain
Word number 269 chomp
Word number 270 restart
Word number 271 restarting
Word number 272 dealers
Word number 273 littered
Word number 274 extenders
Word number 275 extendable
Word number 276 litmus
Word number 277 restocked
Word number 278 literraly
Word number 279 restorations
Word number 280 dean
Word number 281 expulsion
Word number 282 respectively
Word number 283 chokes
Word number 284 liveliness
Word number 285 backside
Word number 286 resiliance
Word number 287 lloyd
Word number 288 badge
Word number 289 lld
Word number 290 lizard
Word number 291 resistably
Word number 292 extracting
Word number 293 resisted
Word number 294 livesnannan
Word number 295 resisting
Word number 296 resistor
Word number 297 livescribe
Word number 298 resolutions
Word number 299 dd1
Word number 300 resolves
Word number 301 extinguish
Word number 302 cotten
Word number 303 extinction
Word number 304 livens
Word number 305 resounding
Word number 306 bacon
Word number 307 backyards
Word number 308 extincted
Word number 309 resourcefulness
Word number 310 backup
Word number 311 choiced
Word number 312 backstory
Word number 313 expressly
Word number 314 restraining

Word number 315 banking
Word number 316 ayn
Word number 317 chordal
Word number 318 choreograph
Word number 319 debilitatingly
Word number 320 retractable
Word number 321 retrain
Word number 322 export
Word number 323 retrieving
Word number 324 retro
Word number 325 retrofit
Word number 326 retry
Word number 327 choreographer
Word number 328 lip
Word number 329 exponents
Word number 330 ayah
Word number 331 restraint
Word number 332 lions
Word number 333 lionni
Word number 334 cot
Word number 335 grammars
Word number 336 explosions
Word number 337 lino
Word number 338 grammer
Word number 339 chormebooks
Word number 340 revelations
Word number 341 grams
Word number 342 reverberation
Word number 343 revere
Word number 344 debrief
Word number 345 rethinking
Word number 346 exposer
Word number 347 retesting
Word number 348 retest
Word number 349 restrict
Word number 350 backburner
Word number 351 dearborn
Word number 352 dearly
Word number 353 restricts
Word number 354 literacies
Word number 355 restrooms
Word number 356 restructure
Word number 357 liter
Word number 358 debarring
Word number 359 backback
Word number 360 bachelor
Word number 361 bache
Word number 362 chopper

```
Word number  363 debatesnannan
Word number  364 resurrecting
Word number  365 listing
Word number  366 listens
Word number  367 listenining
Word number  368 exposition
Word number  369 babysitters
Word number  370 retaking
Word number  371 retardant
Word number  372 debbie
Word number  373 babysitter
Word number  374 babyish
Word number  375 baba
Word number  376 resigned
Word number  377 residue
Word number  378 residing
Word number  379 cougar
Word number  380 repertories
Word number  381 bandaids
Word number  382 gradelevel
Word number  383 bandage
Word number  384 loft
Word number  385 bancroft
Word number  386 replaceable
Word number  387 eyesight
Word number  388 daytime
Word number  389 eyepiece
Word number  390 replay
Word number  391 replaying
Word number  392 bambara
Word number  393 replenishing
Word number  394 extracurriculars
Word number  395 replenishment
Word number  396 lockheed
Word number  397 replicable
Word number  398 lockers
Word number  399 eyeglass
Word number  400 cottonwood
Word number  401 replicates
Word number  402 replicating
Word number  403 lockdown
Word number  404 replies
Word number  405 bam
Word number  406 balto
Word number  407 chineese
Word number  408 bane
Word number  409 repellents
Word number  410 ez
```

Word number 411 loggers
Word number 412 chimamanda
Word number 413 dawned
Word number 414 lonely
Word number 415 lone
Word number 416 bankers
Word number 417 render
Word number 418 rendered
Word number 419 lombardi
Word number 420 renders
Word number 421 rendon
Word number 422 renee
Word number 423 renew
Word number 424 lollipops
Word number 425 daydreaming
Word number 426 lol
Word number 427 lokelani
Word number 428 loins
Word number 429 gradebook
Word number 430 bangs
Word number 431 renta
Word number 432 rented
Word number 433 rents
Word number 434 logistical
Word number 435 reorganize
Word number 436 reorganized
Word number 437 daylight
Word number 438 fables
Word number 439 balsa
Word number 440 repost
Word number 441 balm
Word number 442 baits
Word number 443 resealable
Word number 444 bait
Word number 445 loans
Word number 446 researcher
Word number 447 bagless
Word number 448 researches
Word number 449 dciu
Word number 450 reselling
Word number 451 loaner
Word number 452 resembled
Word number 453 loaned
Word number 454 resentful
Word number 455 baggie
Word number 456 chisholm
Word number 457 chloroplast
Word number 458 reserving

Word number 459 dcps
Word number 460 resettlement
Word number 461 reshapes
Word number 462 lms
Word number 463 bagged
Word number 464 resided
Word number 465 residence
Word number 466 lmfao
Word number 467 resident
Word number 468 lmc
Word number 469 extraordinaires
Word number 470 reseach
Word number 471 rereads
Word number 472 ballroom
Word number 473 extravagant
Word number 474 chinking
Word number 475 ballplayers
Word number 476 extruders
Word number 477 extrude
Word number 478 extrmemly
Word number 479 chip
Word number 480 graditude
Word number 481 chippewa
Word number 482 extrinsically
Word number 483 chipping
Word number 484 reproductions
Word number 485 reps
Word number 486 reptiles
Word number 487 chirping
Word number 488 localities
Word number 489 bald
Word number 490 reputations
Word number 491 extremity
Word number 492 baladacci
Word number 493 dazzlin
Word number 494 bakeware
Word number 495 locales
Word number 496 locale
Word number 497 chisel
Word number 498 baker
Word number 499 baked
Word number 500 extravaganza
Word number 501 lunchboxes
Word number 502 dangers
Word number 503 chester
Word number 504 quanjobal
Word number 505 mally
Word number 506 malls

Word number 507 goofing
Word number 508 googalicous
Word number 509 quaker
Word number 510 qualcomm
Word number 511 mall
Word number 512 qualifiers
Word number 513 malicious
Word number 514 bernoulli
Word number 515 bernardino
Word number 516 berlin
Word number 517 berkner
Word number 518 quantitative
Word number 519 questionable
Word number 520 quantitatively
Word number 521 daddy
Word number 522 feisty
Word number 523 berkeley
Word number 524 mali
Word number 525 daf
Word number 526 quartly
Word number 527 quavermusic
Word number 528 dahle
Word number 529 malfoy
Word number 530 checkbook
Word number 531 queries
Word number 532 malcom
Word number 533 quadraparesis
Word number 534 quadrants
Word number 535 quadcopter
Word number 536 maltreatment
Word number 537 purses
Word number 538 pursing
Word number 539 goodwill
Word number 540 gooey
Word number 541 bethany
Word number 542 beth
Word number 543 goofballs
Word number 544 beta
Word number 545 dabble
Word number 546 felty
Word number 547 pushin
Word number 548 bestselling
Word number 549 pushpins
Word number 550 bestschoolday
Word number 551 putnam
Word number 552 bestowed
Word number 553 cheaply
Word number 554 bess

Word number 555 felted
Word number 556 beseech
Word number 557 puzzling
Word number 558 pwc
Word number 559 pyongyang
Word number 560 pyramids
Word number 561 mammals
Word number 562 mammal
Word number 563 mam
Word number 564 berea
Word number 565 beowulf
Word number 566 feasibility
Word number 567 checkmate
Word number 568 quizmo
Word number 569 quizzed
Word number 570 fedoras
Word number 571 quizzical
Word number 572 quizziz
Word number 573 makery
Word number 574 dalai
Word number 575 googleplex
Word number 576 googler
Word number 577 quoting
Word number 578 quran
Word number 579 qwirkle
Word number 580 r2
Word number 581 raccoons
Word number 582 bents
Word number 583 beneficially
Word number 584 raced
Word number 585 racers
Word number 586 febreze
Word number 587 racetrack
Word number 588 makerbot
Word number 589 bends
Word number 590 feb
Word number 591 checkouts
Word number 592 majors
Word number 593 goop
Word number 594 rackets
Word number 595 maish
Word number 596 checklist
Word number 597 makespace
Word number 598 quizes
Word number 599 googled
Word number 600 bentley
Word number 601 quiche
Word number 602 daigle

Word number 603 bensonhurst
Word number 604 benson
Word number 605 malarkey
Word number 606 quicknet
Word number 607 quickstart
Word number 608 bensalem
Word number 609 dairies
Word number 610 malaga
Word number 611 quietness
Word number 612 checkbooks
Word number 613 quiick
Word number 614 feeders
Word number 615 makin
Word number 616 quilly
Word number 617 quilt
Word number 618 quilts
Word number 619 quimby
Word number 620 quincy
Word number 621 quintessential
Word number 622 quirkiness
Word number 623 googleclassrooms
Word number 624 makeymakey
Word number 625 makeups
Word number 626 quixels
Word number 627 purse
Word number 628 bethune
Word number 629 purposely
Word number 630 fetched
Word number 631 prowler
Word number 632 proxemics
Word number 633 feuds
Word number 634 prs
Word number 635 prune
Word number 636 ps
Word number 637 psi
Word number 638 chatising
Word number 639 pssh
Word number 640 manipulatively
Word number 641 psychiatric
Word number 642 psychical
Word number 643 fete
Word number 644 psychologists
Word number 645 goodnight
Word number 646 chattanooga
Word number 647 psychosis
Word number 648 festivities
Word number 649 bibliotherapy
Word number 650 goldseekers

Word number 651 festive
Word number 652 pts
Word number 653 ptsa
Word number 654 pub
Word number 655 puberty
Word number 656 pubic
Word number 657 bibliothecula
Word number 658 goldsmith
Word number 659 golding
Word number 660 bickering
Word number 661 chasse
Word number 662 provocative
Word number 663 protocol
Word number 664 protocols
Word number 665 fibers
Word number 666 prototyped
Word number 667 goldfishes
Word number 668 prototyping
Word number 669 mantles
Word number 670 fiances
Word number 671 bifidia
Word number 672 mantises
Word number 673 bifida
Word number 674 bieraugel
Word number 675 ffa
Word number 676 bienvenidos
Word number 677 fews
Word number 678 mantids
Word number 679 bidding
Word number 680 bid
Word number 681 bicycling
Word number 682 providence
Word number 683 mant
Word number 684 bicycles
Word number 685 manpulatives
Word number 686 manos
Word number 687 manor
Word number 688 provocation
Word number 689 provocations
Word number 690 manipualtives
Word number 691 biblioteca
Word number 692 bibliophilia
Word number 693 punctuate
Word number 694 ferrero
Word number 695 punctured
Word number 696 punish
Word number 697 punished
Word number 698 chatterpics

Word number 699 goodbyes
Word number 700 fermentation
Word number 701 puppeteering
Word number 702 puppeteers
Word number 703 puppetry
Word number 704 mandating
Word number 705 puppies
Word number 706 puppy
Word number 707 goode
Word number 708 purchasable
Word number 709 bfa
Word number 710 beverly
Word number 711 mandalas
Word number 712 fenton
Word number 713 fences
Word number 714 purge
Word number 715 purification
Word number 716 manannan
Word number 717 purl
Word number 718 chavez
Word number 719 manages
Word number 720 betta
Word number 721 punctuated
Word number 722 punctuality
Word number 723 publisher
Word number 724 punctual
Word number 725 manikins
Word number 726 bibliophiles
Word number 727 puddle
Word number 728 bible
Word number 729 puff
Word number 730 pug
Word number 731 manifested
Word number 732 bibbity
Word number 733 bibbins
Word number 734 manifestation
Word number 735 biases
Word number 736 goobi
Word number 737 pullouts
Word number 738 biased
Word number 739 maniacs
Word number 740 pum
Word number 741 maniac
Word number 742 manhunt
Word number 743 fertility
Word number 744 chatterkid
Word number 745 mangagement
Word number 746 pumps

Word number 747 ferries
Word number 748 punched
Word number 749 maneuvering
Word number 750 mandelareading
Word number 751 mandelai
Word number 752 rad
Word number 753 radiology
Word number 754 fantasies
Word number 755 reblogging
Word number 756 dana
Word number 757 rearranged
Word number 758 macgyver
Word number 759 macfarlane
Word number 760 bedtime
Word number 761 bedrooms
Word number 762 fas
Word number 763 reasources
Word number 764 reassurance
Word number 765 reassure
Word number 766 gotta
Word number 767 reawaken
Word number 768 rebirth
Word number 769 reboot
Word number 770 receivership
Word number 771 macaroni
Word number 772 cherishes
Word number 773 cherokee
Word number 774 rebuttal
Word number 775 rec
Word number 776 goudy
Word number 777 gough
Word number 778 recap
Word number 779 reccess
Word number 780 farmworkers
Word number 781 receipts
Word number 782 bedded
Word number 783 becuase
Word number 784 reared
Word number 785 reapplying
Word number 786 reappear
Word number 787 damper
Word number 788 readinig
Word number 789 mackin
Word number 790 beetles
Word number 791 dampening
Word number 792 beetle
Word number 793 fashioning
Word number 794 readymade

Word number 795 readynannan
Word number 796 reaffirming
Word number 797 reagents
Word number 798 beers
Word number 799 machu
Word number 800 realigned
Word number 801 beeps
Word number 802 realistically
Word number 803 beeper
Word number 804 machining
Word number 805 dampens
Word number 806 chenille
Word number 807 beechfield
Word number 808 beecher
Word number 809 beebot
Word number 810 fasd
Word number 811 cher
Word number 812 realties
Word number 813 machete
Word number 814 maches
Word number 815 lyon
Word number 816 beckon
Word number 817 hiker
Word number 818 reconfigure
Word number 819 lurking
Word number 820 fantastically
Word number 821 beatles
Word number 822 recognizes
Word number 823 fantastical
Word number 824 recollect
Word number 825 lurk
Word number 826 luring
Word number 827 beaters
Word number 828 lured
Word number 829 fantasize
Word number 830 recommends
Word number 831 reconcile
Word number 832 reconnected
Word number 833 beck
Word number 834 lungs
Word number 835 reconstructing
Word number 836 reconstruction
Word number 837 reconvene
Word number 838 beasts
Word number 839 recordable
Word number 840 bearing
Word number 841 bearer
Word number 842 bearden

```
Word number  843 beanstalks
Word number  844 recount
Word number  845 recounted
Word number  846 recoup
Word number  847 beaufort
Word number  848 reclining
Word number  849 reclamation
Word number  850 reclaiming
Word number  851 becasue
Word number  852 lynwood
Word number  853 ly
Word number  854 cheryl
Word number  855 gould
Word number  856 recessed
Word number  857 farmington
Word number  858 recetly
Word number  859 dandelion
Word number  860 goup
Word number  861 recharged
Word number  862 recharging
Word number  863 dandy
Word number  864 lute
Word number  865 farkle
Word number  866 beauties
Word number  867 farewells
Word number  868 fare
Word number  869 lust
Word number  870 reciprocate
Word number  871 reciprocity
Word number  872 recitations
Word number  873 dang
Word number  874 recited
Word number  875 reciting
Word number  876 reckon
Word number  877 reclaim
Word number  878 macleod
Word number  879 readinga
Word number  880 gotalk4
Word number  881 rand
Word number  882 gordan
Word number  883 favors
Word number  884 ramayana
Word number  885 damp
Word number  886 maid
Word number  887 mahalo
Word number  888 mah
Word number  889 magnus
Word number  890 rampshot
```

```
Word number  891 rams
Word number  892 magnitude
Word number  893 ranches
Word number  894 magnify
Word number  895 randall
Word number  896 macroinvertebrates
Word number  897 randolph
Word number  898 favored
Word number  899 randomized
Word number  900 magnifiers
Word number  901 rang
Word number  902 belorussia
Word number  903 ranged
Word number  904 magnified
Word number  905 gorge
Word number  906 cheesy
Word number  907 cheetah
Word number  908 rankings
Word number  909 faulty
Word number  910 goprso
Word number  911 rakes
Word number  912 fcat
Word number  913 mainstays
Word number  914 radish
Word number  915 fearlessly
Word number  916 raffled
Word number  917 raffles
Word number  918 raft
Word number  919 rag
Word number  920 rags
Word number  921 raider
Word number  922 raiders
Word number  923 raids
Word number  924 rail
Word number  925 fearfully
Word number  926 gopigo
Word number  927 fearful
Word number  928 damion
Word number  929 mainstreaming
Word number  930 raindrop
Word number  931 rained
Word number  932 gopro
Word number  933 raining
Word number  934 fe
Word number  935 rainwater
Word number  936 cheerleader
Word number  937 beluga
Word number  938 beltway
```

```
Word number  939 raiser
Word number  940 raisers
Word number  941 rap
Word number  942 rape
Word number  943 raphael
Word number  944 magee
Word number  945 chemist
Word number  946 chemists
Word number  947 fatalities
Word number  948 mag
Word number  949 rbms
Word number  950 rc
Word number  951 rdspd
Word number  952 re
Word number  953 behemoths
Word number  954 behaviours
Word number  955 behaviorthe
Word number  956 behaviorial
Word number  957 maduros
Word number  958 behaves
Word number  959 fastt
Word number  960 gotalk
Word number  961 reacts
Word number  962 chen
Word number  963 read180
Word number  964 madden
Word number  965 mact
Word number  966 readasaurus
Word number  967 fastly
Word number  968 dampeners
Word number  969 macroscopic
Word number  970 macros
Word number  971 readin
Word number  972 rawls
Word number  973 magenta
Word number  974 bellringer
Word number  975 ravenous
Word number  976 faulted
Word number  977 rapp
Word number  978 rapping
Word number  979 faulkner
Word number  980 magnatism
Word number  981 bellevue
Word number  982 belle
Word number  983 faucets
Word number  984 faucet
Word number  985 gorgeous
Word number  986 ratchet
```

```
Word number  987 fattening
Word number  988 magician
Word number  989 ratey
Word number  990 believ
Word number  991 gossip
Word number  992 fathomable
Word number  993 counsels
Word number  994 ration
Word number  995 rational
Word number  996 rationale
Word number  997 rationals
Word number  998 chem
Word number  999 raton
Word number  1000 maggie
Word number  1001 ravage
Word number  1002 magestic
Word number  1003 reversing
Word number  1004 debriefing
Word number  1005 goldenrod
Word number  1006 selfesteem
Word number  1007 everythinv
Word number  1008 classesnannan
Word number  1009 definitively
Word number  1010 deflate
Word number  1011 classflexible
Word number  1012 evergreen
Word number  1013 languag
Word number  1014 ashley
Word number  1015 ashamed
Word number  1016 everest
Word number  1017 lane
Word number  1018 landscaping
Word number  1019 ash
Word number  1020 selfie
Word number  1021 senario
Word number  1022 selfies
Word number  1023 greensboro
Word number  1024 landscapers
Word number  1025 ascii
Word number  1026 sellers
Word number  1027 deflect
Word number  1028 selphy
Word number  1029 selpy
Word number  1030 ascent
Word number  1031 landsat
Word number  1032 ascd
Word number  1033 deforestation
Word number  1034 semicircles
```

```
Word number  1035 seismographs
Word number  1036 seismograph
Word number  1037 seismic
Word number  1038 everytime
Word number  1039 sedimentarty
Word number  1040 sedimentary
Word number  1041 deficiency
Word number  1042 assemblages
Word number  1043 evoking
Word number  1044 assemblage
Word number  1045 aspires
Word number  1046 aspired
Word number  1047 evoke
Word number  1048 lapping
Word number  1049 aspirantes
Word number  1050 lapboards
Word number  1051 evocative
Word number  1052 seeming
Word number  1053 greenness
Word number  1054 aspen
Word number  1055 lanyards
Word number  1056 lanuage
Word number  1057 asled
Word number  1058 seetouchlearn
Word number  1059 segmentation
Word number  1060 segmented
Word number  1061 segmenting
Word number  1062 lanning
Word number  1063 lanier
Word number  1064 languange
Word number  1065 segue
Word number  1066 defunding
Word number  1067 senate
Word number  1068 lapse
Word number  1069 servers
Word number  1070 greta
Word number  1071 articulating
Word number  1072 dehydrated
Word number  1073 articulated
Word number  1074 classifies
Word number  1075 ser
Word number  1076 serena
Word number  1077 corrective
Word number  1078 serine
Word number  1079 artful
Word number  1080 serotonin
Word number  1081 artemis
Word number  1082 arte
```

Word number 1083 arsty
Word number 1084 asbestos
Word number 1085 arsenal
Word number 1086 evacuate
Word number 1087 arrrgh
Word number 1088 classism
Word number 1089 arrows
Word number 1090 servings
Word number 1091 servo
Word number 1092 classroomnannan
Word number 1093 lamborghini
Word number 1094 euro
Word number 1095 arrest
Word number 1096 setback
Word number 1097 lambeth
Word number 1098 articulations
Word number 1099 separations
Word number 1100 classifications
Word number 1101 lana
Word number 1102 gregarious
Word number 1103 asa
Word number 1104 gregor
Word number 1105 senor
Word number 1106 senosory
Word number 1107 degradation
Word number 1108 eve
Word number 1109 landers
Word number 1110 evaporation
Word number 1111 senseez
Word number 1112 artset
Word number 1113 sensing
Word number 1114 artsedge
Word number 1115 evaporates
Word number 1116 evanston
Word number 1117 evan
Word number 1118 artistry
Word number 1119 sensual
Word number 1120 evaluator
Word number 1121 landed
Word number 1122 degrasse
Word number 1123 classification
Word number 1124 sentiments
Word number 1125 artifact
Word number 1126 lancaster
Word number 1127 separately
Word number 1128 separates
Word number 1129 assembled
Word number 1130 lapsed

Word number 1131 axl
Word number 1132 laughlin
Word number 1133 excelence
Word number 1134 laundromat
Word number 1135 scrapbook
Word number 1136 scrapbooks
Word number 1137 scrape
Word number 1138 scraper
Word number 1139 scrapes
Word number 1140 launchpad
Word number 1141 scrapping
Word number 1142 scratchboard
Word number 1143 launched
Word number 1144 clamor
Word number 1145 clamoring
Word number 1146 screeching
Word number 1147 scrimped
Word number 1148 defeating
Word number 1149 screenchomp
Word number 1150 clamp
Word number 1151 clamped
Word number 1152 screenplays
Word number 1153 lauded
Word number 1154 screenshots
Word number 1155 screw
Word number 1156 latter
Word number 1157 scribble
Word number 1158 scribbler
Word number 1159 scribes
Word number 1160 scribing
Word number 1161 launguage
Word number 1162 scrabble
Word number 1163 scouring
Word number 1164 laurence
Word number 1165 sciencewiz
Word number 1166 greatfloodof2016
Word number 1167 deere
Word number 1168 greatfloodofds2016
Word number 1169 layering
Word number 1170 scientistsi
Word number 1171 scigirls
Word number 1172 scintillating
Word number 1173 scissor
Word number 1174 scoical
Word number 1175 layaway
Word number 1176 claiming
Word number 1177 assumptions
Word number 1178 greco

```
Word number  1179 excellency
Word number  1180 lawrence
Word number  1181 lawns
Word number  1182 scootpad
Word number  1183 lawndale
Word number  1184 scorching
Word number  1185 greedy
Word number  1186 scorebooks
Word number  1187 lavelle
Word number  1188 lava
Word number  1189 assuage
Word number  1190 lausd
Word number  1191 scotch
Word number  1192 latte
Word number  1193 scrimping
Word number  1194 clasroom
Word number  1195 sec
Word number  1196 defiant
Word number  1197 seasonally
Word number  1198 seasoned
Word number  1199 greenery
Word number  1200 defibrillator
Word number  1201 seatin
Word number  1202 assesses
Word number  1203 seatings
Word number  1204 larson
Word number  1205 clark
Word number  1206 seatwork
Word number  1207 sebastian
Word number  1208 sebastopol
Word number  1209 secluded
Word number  1210 excavating
Word number  1211 eww
Word number  1212 clarkston
Word number  1213 ewriters
Word number  1214 assertiveness
Word number  1215 clas
Word number  1216 secretaries
Word number  1217 secretary
Word number  1218 secretly
Word number  1219 secrets
Word number  1220 larchmont
Word number  1221 lara
Word number  1222 sector
Word number  1223 secular
Word number  1224 assimilating
Word number  1225 seas
Word number  1226 searing
```

Word number 1227 lasseter
Word number 1228 latinx
Word number 1229 associates
Word number 1230 scriptwriting
Word number 1231 clanton
Word number 1232 scrounged
Word number 1233 scrounging
Word number 1234 scrub
Word number 1235 scrutinizing
Word number 1236 scuffles
Word number 1237 excavate
Word number 1238 sculpted
Word number 1239 lathe
Word number 1240 clapping
Word number 1241 sculptural
Word number 1242 defenders
Word number 1243 latent
Word number 1244 greenberg
Word number 1245 sdc
Word number 1246 clarifications
Word number 1247 examined
Word number 1248 seahorses
Word number 1249 latches
Word number 1250 seals
Word number 1251 greene
Word number 1252 clarinetist
Word number 1253 assistances
Word number 1254 lasteat
Word number 1255 seth
Word number 1256 arrangments
Word number 1257 setter
Word number 1258 shovel
Word number 1259 shortly
Word number 1260 archetypes
Word number 1261 shotgun
Word number 1262 escalating
Word number 1263 escalate
Word number 1264 kristi
Word number 1265 erupt
Word number 1266 ert
Word number 1267 krista
Word number 1268 krazy
Word number 1269 krafty
Word number 1270 shove
Word number 1271 kpa
Word number 1272 kozy
Word number 1273 ernest
Word number 1274 kozol

Word number 1275 archetype
Word number 1276 arches
Word number 1277 ers
Word number 1278 koval
Word number 1279 deltamath
Word number 1280 korman
Word number 1281 koreatown
Word number 1282 grimms
Word number 1283 grin
Word number 1284 clearance
Word number 1285 archaic
Word number 1286 shred
Word number 1287 escaped
Word number 1288 kristopher
Word number 1289 escorting
Word number 1290 shortened
Word number 1291 shmoop
Word number 1292 kurzweil
Word number 1293 kurt
Word number 1294 kumari
Word number 1295 shockproof
Word number 1296 delineate
Word number 1297 shoelaces
Word number 1298 archived
Word number 1299 shoeshine
Word number 1300 kudos
Word number 1301 shook
Word number 1302 archive
Word number 1303 especiallthese
Word number 1304 delineated
Word number 1305 kti
Word number 1306 eslcafe
Word number 1307 deliveries
Word number 1308 shoreline
Word number 1309 architectual
Word number 1310 architect
Word number 1311 shortchanged
Word number 1312 krosoczka
Word number 1313 escuela
Word number 1314 shortcut
Word number 1315 krop
Word number 1316 shorted
Word number 1317 shorten
Word number 1318 koosh
Word number 1319 shredding
Word number 1320 eurhythmics
Word number 1321 sightseeing
Word number 1322 clearning

Word number 1323 sidestep
Word number 1324 sidetracked
Word number 1325 erik
Word number 1326 sidewalks
Word number 1327 siempre
Word number 1328 clears
Word number 1329 grindz
Word number 1330 sigh
Word number 1331 arabics
Word number 1332 sighted
Word number 1333 sightly
Word number 1334 erie
Word number 1335 knowldge
Word number 1336 shrink
Word number 1337 sightwords
Word number 1338 sighword
Word number 1339 arab
Word number 1340 signage
Word number 1341 clef
Word number 1342 signalling
Word number 1343 knots
Word number 1344 ergonomics
Word number 1345 knostthe
Word number 1346 deltora
Word number 1347 aquisition
Word number 1348 aquire
Word number 1349 grins
Word number 1350 knowledgenannan
Word number 1351 sidekicks
Word number 1352 sidekick
Word number 1353 erika
Word number 1354 eritrean
Word number 1355 eritrea
Word number 1356 shrubs
Word number 1357 shrunk
Word number 1358 shudder
Word number 1359 shuffle
Word number 1360 shuffled
Word number 1361 kokomo
Word number 1362 shufflin
Word number 1363 kohlberg
Word number 1364 shun
Word number 1365 shunned
Word number 1366 shusterman
Word number 1367 kohl
Word number 1368 kobe
Word number 1369 koala
Word number 1370 shuttle

Word number 1371 shuttlecocks
Word number 1372 shuttles
Word number 1373 archaeological
Word number 1374 shyness
Word number 1375 grinch
Word number 1376 arcf
Word number 1377 clearing
Word number 1378 sicken
Word number 1379 sickening
Word number 1380 arboretum
Word number 1381 archives
Word number 1382 archon
Word number 1383 shipyards
Word number 1384 shallower
Word number 1385 grey
Word number 1386 shadowed
Word number 1387 shadowing
Word number 1388 lags
Word number 1389 ethiopia
Word number 1390 shaker
Word number 1391 lafs
Word number 1392 lafollette
Word number 1393 shakespearean
Word number 1394 shakin
Word number 1395 ethernet
Word number 1396 shalf
Word number 1397 etd
Word number 1398 shalom
Word number 1399 grill
Word number 1400 lafayette
Word number 1401 griddle
Word number 1402 classsroom
Word number 1403 shannon
Word number 1404 arista
Word number 1405 arising
Word number 1406 shapers
Word number 1407 ladibug
Word number 1408 shapeshifters
Word number 1409 etch
Word number 1410 sharan
Word number 1411 delectables
Word number 1412 shareable
Word number 1413 shading
Word number 1414 delaying
Word number 1415 laika
Word number 1416 delaware
Word number 1417 lamanna
Word number 1418 eureka

Word number 1419 settingupforsecond
Word number 1420 euphonium
Word number 1421 euless
Word number 1422 settlement
Word number 1423 settlers
Word number 1424 lama
Word number 1425 lam
Word number 1426 arne
Word number 1427 seussthe
Word number 1428 seussville
Word number 1429 euler
Word number 1430 etrex
Word number 1431 lakota
Word number 1432 armed
Word number 1433 lakewood
Word number 1434 etite
Word number 1435 arius
Word number 1436 severities
Word number 1437 dekalb
Word number 1438 sewer
Word number 1439 sewn
Word number 1440 lakeland
Word number 1441 sexes
Word number 1442 classrrom
Word number 1443 sexual
Word number 1444 ariana
Word number 1445 laden
Word number 1446 argumentation
Word number 1447 areading
Word number 1448 shelve
Word number 1449 labeler
Word number 1450 deleted
Word number 1451 shenandoah
Word number 1452 shenanigans
Word number 1453 shepard
Word number 1454 sheppard
Word number 1455 sherman
Word number 1456 sheryl
Word number 1457 shevet
Word number 1458 shh
Word number 1459 shhh
Word number 1460 shhhhh
Word number 1461 shidle
Word number 1462 deliberate
Word number 1463 ardent
Word number 1464 est
Word number 1465 essex
Word number 1466 essentle

Word number 1467 shimabukuro
Word number 1468 arctic
Word number 1469 shined
Word number 1470 kyptonite
Word number 1471 shinguards
Word number 1472 delicate
Word number 1473 kyle
Word number 1474 essdee
Word number 1475 cle
Word number 1476 shelly
Word number 1477 lacrosse
Word number 1478 lables
Word number 1479 necesitamos
Word number 1480 classworks
Word number 1481 sharpe
Word number 1482 estimated
Word number 1483 argued
Word number 1484 griddles
Word number 1485 claude
Word number 1486 ares
Word number 1487 labsheets
Word number 1488 claus
Word number 1489 sharpie
Word number 1490 esteems
Word number 1491 sharpness
Word number 1492 sharpy
Word number 1493 shasta
Word number 1494 labreadoodle
Word number 1495 shavings
Word number 1496 shcs
Word number 1497 laborious
Word number 1498 esteemed
Word number 1499 sheen
Word number 1500 aremostly
Word number 1501 aregular
Word number 1502 areally
Word number 1503 shel
Word number 1504 laborers
Word number 1505 delete
Word number 1506 astill
Word number 1507 astonish
Word number 1508 schushterman
Word number 1509 automata
Word number 1510 expenditure
Word number 1511 rockaway
Word number 1512 licensed
Word number 1513 expended
Word number 1514 chuckle

```
Word number  1515 rocketed
Word number  1516 rocketry
Word number  1517 expelled
Word number  1518 licences
Word number  1519 libya
Word number  1520 rockmath
Word number  1521 rockout
Word number  1522 rockridge
Word number  1523 rockstar
Word number  1524 chunking
Word number  1525 expedited
Word number  1526 rockwell
Word number  1527 chugga
Word number  1528 rode
Word number  1529 chugiak
Word number  1530 libraray
Word number  1531 liberation
Word number  1532 liberate
Word number  1533 rokenbok
Word number  1534 autocad
Word number  1535 autobiographies
Word number  1536 liable
Word number  1537 lia
Word number  1538 granting
Word number  1539 roche
Word number  1540 licious
Word number  1541 robtics
Word number  1542 rivera
Word number  1543 decks
Word number  1544 riverside
Word number  1545 declared
Word number  1546 chrysanthemum
Word number  1547 ro
Word number  1548 roaches
Word number  1549 lifeline
Word number  1550 experiements
Word number  1551 lifecycle
Word number  1552 roadways
Word number  1553 experieces
Word number  1554 chs
Word number  1555 roar
Word number  1556 expereinces
Word number  1557 chuckey
Word number  1558 robins
Word number  1559 robinson
Word number  1560 cospaces
Word number  1561 automobile
Word number  1562 robotc
```

Word number 1563 automatize
Word number 1564 robotically
Word number 1565 lid
Word number 1566 roboting
Word number 1567 expereince
Word number 1568 robs
Word number 1569 lgpe
Word number 1570 autisim
Word number 1571 lettercising
Word number 1572 augustine
Word number 1573 graphically
Word number 1574 ross
Word number 1575 expec
Word number 1576 rosters
Word number 1577 rotary
Word number 1578 aussie
Word number 1579 chunky
Word number 1580 levelizes
Word number 1581 expansive
Word number 1582 cosmetologists
Word number 1583 decompress
Word number 1584 rotorcraft
Word number 1585 cory
Word number 1586 rougher
Word number 1587 cosmic
Word number 1588 roughest
Word number 1589 roughhousing
Word number 1590 augmenting
Word number 1591 leve
Word number 1592 deconstructing
Word number 1593 exotic
Word number 1594 roundups
Word number 1595 graphite
Word number 1596 exorbitant
Word number 1597 routers
Word number 1598 exodus
Word number 1599 churro
Word number 1600 exiting
Word number 1601 rosenstock
Word number 1602 rosemont
Word number 1603 rosas
Word number 1604 levine
Word number 1605 rom
Word number 1606 lfp
Word number 1607 romania
Word number 1608 leyson
Word number 1609 romare
Word number 1610 lexiles

Word number 1611 lexicore5
Word number 1612 ronaldo
Word number 1613 cosmetology
Word number 1614 graphical
Word number 1615 roomful
Word number 1616 roomier
Word number 1617 rooming
Word number 1618 authoring
Word number 1619 lewin
Word number 1620 levy
Word number 1621 rooseveltthese
Word number 1622 rooster
Word number 1623 roosters
Word number 1624 decomposers
Word number 1625 levitt
Word number 1626 levitation
Word number 1627 authenticates
Word number 1628 levitating
Word number 1629 roped
Word number 1630 expecience
Word number 1631 roping
Word number 1632 lifes
Word number 1633 rivals
Word number 1634 rivalries
Word number 1635 chromecast
Word number 1636 christinabhoward
Word number 1637 rezoned
Word number 1638 rf
Word number 1639 rfqqh7iccounannan
Word number 1640 rgb
Word number 1641 explode
Word number 1642 rhetorical
Word number 1643 rhino
Word number 1644 rhinoskin
Word number 1645 chromatic
Word number 1646 grander
Word number 1647 aways
Word number 1648 awarding
Word number 1649 limestone
Word number 1650 lifesavers
Word number 1651 explicate
Word number 1652 decay
Word number 1653 ric
Word number 1654 limeadesforlearning
Word number 1655 ricans
Word number 1656 deceased
Word number 1657 deception
Word number 1658 richard

Word number 1659 avon
Word number 1660 riches
Word number 1661 lime
Word number 1662 richland
Word number 1663 limbo
Word number 1664 rex
Word number 1665 lindblom
Word number 1666 linden
Word number 1667 christi
Word number 1668 axis
Word number 1669 awoken
Word number 1670 linguists
Word number 1671 debris
Word number 1672 revision
Word number 1673 choruses
Word number 1674 chris
Word number 1675 revisited
Word number 1676 lingering
Word number 1677 linger
Word number 1678 revitalized
Word number 1679 revitalizing
Word number 1680 debug
Word number 1681 explorative
Word number 1682 debut
Word number 1683 debuted
Word number 1684 lingala
Word number 1685 revolutions
Word number 1686 exploitative
Word number 1687 exploding
Word number 1688 christa
Word number 1689 revved
Word number 1690 christenberry
Word number 1691 awfully
Word number 1692 awful
Word number 1693 awestruck
Word number 1694 rewind
Word number 1695 expirence
Word number 1696 lilo
Word number 1697 lilly
Word number 1698 granice
Word number 1699 ringgggg
Word number 1700 ringing
Word number 1701 ringleader
Word number 1702 avengers
Word number 1703 rio
Word number 1704 ligatures
Word number 1705 riots
Word number 1706 experientially

Word number 1707 riparian
Word number 1708 ripe
Word number 1709 ripley
Word number 1710 ave
Word number 1711 ligature
Word number 1712 granite
Word number 1713 avant
Word number 1714 risell
Word number 1715 lifetimes
Word number 1716 rises
Word number 1717 chronicling
Word number 1718 autopsy
Word number 1719 chrysalis
Word number 1720 rit
Word number 1721 decker
Word number 1722 lifeskills
Word number 1723 rithmatic
Word number 1724 riting
Word number 1725 lifeskill
Word number 1726 ringers
Word number 1727 grandview
Word number 1728 rickety
Word number 1729 rims
Word number 1730 ricki
Word number 1731 expetations
Word number 1732 deceptively
Word number 1733 chromed
Word number 1734 avoidance
Word number 1735 rider
Word number 1736 decibel
Word number 1737 decibels
Word number 1738 ridged
Word number 1739 cosumnes
Word number 1740 liives
Word number 1741 ridiculing
Word number 1742 ridiculously
Word number 1743 lightspeed
Word number 1744 rif
Word number 1745 rific
Word number 1746 rifles
Word number 1747 aviation
Word number 1748 chromosomal
Word number 1749 aversion
Word number 1750 chromosome
Word number 1751 lighter
Word number 1752 averaging
Word number 1753 lightened
Word number 1754 decidedly

Word number 1755 decimated
Word number 1756 lightbulbs
Word number 1757 aug
Word number 1758 churros
Word number 1759 schulwerk
Word number 1760 ats
Word number 1761 satiate
Word number 1762 sational
Word number 1763 satirical
Word number 1764 attachment
Word number 1765 learningmy
Word number 1766 attaching
Word number 1767 learningchromebooks
Word number 1768 sats
Word number 1769 learniing
Word number 1770 learnersnannan
Word number 1771 circus
Word number 1772 learnermy
Word number 1773 circut
Word number 1774 cis
Word number 1775 leapreader
Word number 1776 learnable
Word number 1777 atrocities
Word number 1778 excitrment
Word number 1779 savories
Word number 1780 savoring
Word number 1781 atp
Word number 1782 savy
Word number 1783 atorium
Word number 1784 sawing
Word number 1785 saws
Word number 1786 deduction
Word number 1787 atomsphere
Word number 1788 sayin
Word number 1789 satellite
Word number 1790 sarcasm
Word number 1791 graves
Word number 1792 learnzillion
Word number 1793 attacks
Word number 1794 sanctioned
Word number 1795 sanctity
Word number 1796 circulated
Word number 1797 attacking
Word number 1798 lecroy
Word number 1799 sanders
Word number 1800 sanderson
Word number 1801 lebourgeois
Word number 1802 sands

Word number 1803 leblond
Word number 1804 lebanon
Word number 1805 sandwiches
Word number 1806 leavening
Word number 1807 sane
Word number 1808 executes
Word number 1809 executed
Word number 1810 sanitation
Word number 1811 excursions
Word number 1812 circumference
Word number 1813 sanity
Word number 1814 sanjay
Word number 1815 sansa
Word number 1816 leas
Word number 1817 santiago
Word number 1818 sapce
Word number 1819 saquen
Word number 1820 leapreaders
Word number 1821 corroborate
Word number 1822 roving
Word number 1823 scholastics
Word number 1824 ata
Word number 1825 leach
Word number 1826 lea
Word number 1827 greater4
Word number 1828 lb
Word number 1829 schlosser
Word number 1830 astronomical
Word number 1831 excepted
Word number 1832 astronomer
Word number 1833 excels
Word number 1834 astronaut
Word number 1835 astrology
Word number 1836 lazyboy
Word number 1837 astrobiology
Word number 1838 sba
Word number 1839 schoolboys
Word number 1840 correspondences
Word number 1841 astraunat
Word number 1842 schoolflexible
Word number 1843 lays
Word number 1844 schoolmates
Word number 1845 schoolnannan
Word number 1846 schoolnet
Word number 1847 schoology
Word number 1848 schoolpad
Word number 1849 layouts
Word number 1850 schoolthe

Word number 1851 astonishing
Word number 1852 scents
Word number 1853 scented
Word number 1854 scent
Word number 1855 atalas
Word number 1856 leans
Word number 1857 atleast
Word number 1858 gravitated
Word number 1859 deed
Word number 1860 leaner
Word number 1861 scafolding
Word number 1862 atlases
Word number 1863 atlas
Word number 1864 citement
Word number 1865 deeds
Word number 1866 excitability
Word number 1867 leaky
Word number 1868 leaking
Word number 1869 scant
Word number 1870 scarcely
Word number 1871 citrus
Word number 1872 athletically
Word number 1873 deepened
Word number 1874 excessively
Word number 1875 scatter
Word number 1876 excersice
Word number 1877 civ
Word number 1878 corresponds
Word number 1879 excercising
Word number 1880 athe
Word number 1881 excepts
Word number 1882 deepens
Word number 1883 dedicates
Word number 1884 lecturer
Word number 1885 exemplified
Word number 1886 attuned
Word number 1887 leoni
Word number 1888 leon
Word number 1889 cortices
Word number 1890 audacity
Word number 1891 cincinnati
Word number 1892 ruling
Word number 1893 rum
Word number 1894 lenovo
Word number 1895 rummaging
Word number 1896 rumor
Word number 1897 exhilarate
Word number 1898 lenoir

Word number 1899 lengthening
Word number 1900 cinco
Word number 1901 sampled
Word number 1902 runways
Word number 1903 cortical
Word number 1904 attributed
Word number 1905 cinema
Word number 1906 rushes
Word number 1907 exhibitions
Word number 1908 rushmore
Word number 1909 russell
Word number 1910 cinematic
Word number 1911 exhibiting
Word number 1912 lemons
Word number 1913 rusted
Word number 1914 rustling
Word number 1915 ruins
Word number 1916 leotards
Word number 1917 exile
Word number 1918 audiory
Word number 1919 rovs
Word number 1920 letsmove
Word number 1921 rowell
Word number 1922 rower
Word number 1923 lethargy
Word number 1924 lest
Word number 1925 ciao
Word number 1926 rqlrucfor
Word number 1927 rr
Word number 1928 rriculum
Word number 1929 rs
Word number 1930 cic
Word number 1931 rt
Word number 1932 audiovisuals
Word number 1933 rub
Word number 1934 audiovisual
Word number 1935 rubberbanding
Word number 1936 rubbermaid
Word number 1937 corvallis
Word number 1938 cicero
Word number 1939 rube
Word number 1940 rubiks
Word number 1941 rubix
Word number 1942 cilantro
Word number 1943 leses
Word number 1944 ruby
Word number 1945 rudimentary
Word number 1946 exhibited

Word number 1947 rut
Word number 1948 ruts
Word number 1949 legacies
Word number 1950 sailor
Word number 1951 sails
Word number 1952 decorator
Word number 1953 saints
Word number 1954 attendees
Word number 1955 attenborough
Word number 1956 salads
Word number 1957 grassley
Word number 1958 exerciser
Word number 1959 leftovers
Word number 1960 grassy
Word number 1961 salem
Word number 1962 circular
Word number 1963 leeway
Word number 1964 salespeople
Word number 1965 leeward
Word number 1966 leech
Word number 1967 salon
Word number 1968 salsas
Word number 1969 ledger
Word number 1970 salut
Word number 1971 exercised
Word number 1972 salvation
Word number 1973 samcam
Word number 1974 exemplifying
Word number 1975 gratefully
Word number 1976 gratefulness
Word number 1977 sailing
Word number 1978 grasslands
Word number 1979 rves
Word number 1980 saginaw
Word number 1981 rye
Word number 1982 sa
Word number 1983 saavy
Word number 1984 saber
Word number 1985 sabinrobotics
Word number 1986 sachs
Word number 1987 leisurenannan
Word number 1988 cinematography
Word number 1989 leinkauf
Word number 1990 sacred
Word number 1991 exhaustion
Word number 1992 exhaling
Word number 1993 sacrificing
Word number 1994 attracting

```
Word number  1995 cinnamon
Word number  1996 circe
Word number  1997 saddle
Word number  1998 legitimate
Word number  1999 exertion
```

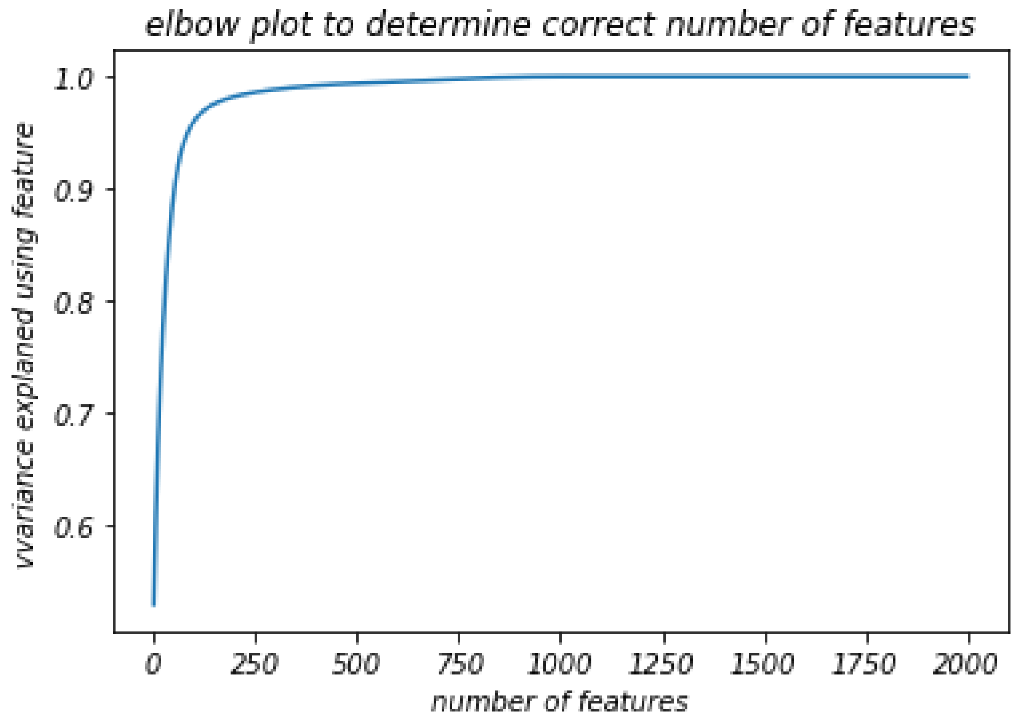
2.3 Applying TruncatedSVD and Calculating Vectors for `essay` and `project_title`

```
In [84]: print(cooc_train.shape)
```

```
(2000, 2000)
```

```
In [85]: #https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD
         from sklearn.decomposition import TruncatedSVD
         svd = TruncatedSVD(n_components=1999, random_state=42)
         svd.fit(cooc_train)
         print(svd.explained_variance_ratio_)
         cumulative_sum=np.cumsum(svd.explained_variance_ratio_)
         a=np.arange(1,2000)
         plt.plot(a,cumulative_sum)
         plt.xlabel('number of features')
         plt.ylabel('vvariance explaned using feature')
         plt.title('elbow plot to determine correct number of features')
         plt.show()
```

```
[5.29492308e-01 1.86918086e-02 1.67976827e-02 ... 2.78954651e-47
 1.00457874e-47 8.54378760e-48]
```




```
In [87]: from sklearn.decomposition import TruncatedSVD
         svd = TruncatedSVD(n_components=250, random_state=42)
         final_w2v= svd.fit_transform(cooc_train)
         print(final_w2v.shape)

(2000, 250)


In [89]: dictionary_w2v={}
         for i,j in enumerate(top2000_words):
             dictionary_w2v[j]=final_w2v[i,:]

In [93]: # average Word2Vec
         # compute average word2vec for each review.
         avg_w2v_vectors_combined_essay_title_train = []; # the avg-w2v for each sentence/revi
         for sentence in tqdm(project_data_X_train.combined_essay_title): # for each review/se
             vector = np.zeros(250) # 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 dictionary_w2v:
                     vector += dictionary_w2v[word]
                     cnt_words += 1
             if cnt_words != 0:
                 vector /= cnt_words
```

```
        avg_w2v_vectors_combined_essay_title_train.append(vector)

    print(len(avg_w2v_vectors_combined_essay_title_train))
    print(len(avg_w2v_vectors_combined_essay_title_train[0]))
```

```
100%|| 8000/8000 [00:00<00:00, 39514.43it/s]

8000
250
```

In [95]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_combined_essay_title_test = []; # the avg-w2v for each sentence/revie
for sentence in tqdm(project_data_X_test.essay.values): # for each review/sentence
    vector = np.zeros(250) # 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 dictionary_w2v:
            vector += dictionary_w2v[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_combined_essay_title_test.append(vector)

print(len(avg_w2v_vectors_combined_essay_title_test))
print(len(avg_w2v_vectors_combined_essay_title_test[0]))
```

```
100%|| 2000/2000 [00:00<00:00, 41776.56it/s]

2000
250
```

2.4 Merge the features from step 3 and step 4
2.4.1 Categorical features

In [96]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_clean_categories = CountVectorizer(vocabulary=list(sorted_cat_dict.keys())
vectorizer_clean_categories.fit(project_data_X_train['clean_categories'].values)
print(vectorizer_clean_categories.get_feature_names())

#for train data
categories_one_hot_train = vectorizer_clean_categories.transform(project_data_X_train
print("Shape of matrix after one hot encodig ",categories_one_hot_train.shape)
```

```
#for test
categories_one_hot_test = vectorizer_clean_categories.transform(project_data_X_test['c
print("Shape of matrix after one hot encodig ",categories_one_hot_test.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'l
Shape of matrix after one hot encodig  (8000, 9)
Shape of matrix after one hot encodig  (2000, 9)
```

In [97]:
```
vectorizer_clean_subcategories = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.k
vectorizer_clean_subcategories.fit(project_data_X_train['clean_subcategories'].values)
print(vectorizer_clean_subcategories.get_feature_names())

#for train data
sub_categories_one_hot_train = vectorizer_clean_subcategories.transform(project_data_X
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_train.shape)

#for test
sub_categories_one_hot_test = vectorizer_clean_subcategories.transform(project_data_X_
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_test.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
Shape of matrix after one hot encodig  (8000, 30)
Shape of matrix after one hot encodig  (2000, 30)
```

In [98]:
```
project_data_X_train.teacher_prefix = project_data_X_train.teacher_prefix.replace(np.n
print(project_data_X_train.teacher_prefix.value_counts())
project_data_X_test.teacher_prefix = project_data_X_test.teacher_prefix.replace(np.nan
print(project_data_X_test.teacher_prefix.value_counts())
```

```
Mrs.       4264
Ms.        2819
Mr.         762
Teacher     155
Name: teacher_prefix, dtype: int64
Mrs.       1103
Ms.         651
Mr.         194
Teacher      52
Name: teacher_prefix, dtype: int64
```

In [99]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_teacher_prefix = CountVectorizer(vocabulary=['Mrs.','Ms.','Mr.','Teacher',
vectorizer_teacher_prefix.fit(project_data_X_train['teacher_prefix'].values)
print(vectorizer_teacher_prefix.get_feature_names())

teacher_prefix_one_hot_train = vectorizer_teacher_prefix.transform(project_data_X_tra
```

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

teacher_prefix_one_hot_test = vectorizer_teacher_prefix.transform(project_data_X_test
print("Shape of matrix after one hot encodig ",teacher_prefix_one_hot_test.shape)
```

```
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (8000, 5)
Shape of matrix after one hot encodig  (2000, 5)
```

In [100]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_project_grade_category = CountVectorizer(vocabulary=list(project_data_X_tr
vectorizer_project_grade_category.fit(project_data_X_train['project_grade_category']
print(vectorizer_project_grade_category.get_feature_names())

project_grade_category_one_hot_train = vectorizer_project_grade_category.transform(pr
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_train.s

project_grade_category_one_hot_test = vectorizer_project_grade_category.transform(pro
print("Shape of matrix after one hot encodig ",project_grade_category_one_hot_test.sh
```

```
['Grades 6-8', 'Grades 9-12', 'Grades PreK-2', 'Grades 3-5']
Shape of matrix after one hot encodig  (8000, 4)
Shape of matrix after one hot encodig  (2000, 4)
```

In [101]:
```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_school_state = CountVectorizer(vocabulary=list(project_data_X_train['schoo
vectorizer_school_state.fit(project_data_X_train['school_state'].values)
print(vectorizer_school_state.get_feature_names())


school_state_one_hot_train = vectorizer_school_state.transform(project_data_X_train[
print("Shape of matrix after one hot encodig ",school_state_one_hot_train.shape)

school_state_one_hot_test = vectorizer_school_state.transform(project_data_X_test['sc
print("Shape of matrix after one hot encodig ",school_state_one_hot_test.shape)
```

```
['MD', 'NH', 'NY', 'SC', 'CA', 'TX', 'OK', 'MA', 'NE', 'NC', 'OH', 'UT', 'MS', 'NJ', 'MI', 'GA
Shape of matrix after one hot encodig  (8000, 51)
Shape of matrix after one hot encodig  (2000, 51)
```

2.2.2 Numerical features

In [102]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn
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.   .
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data_X_train['price'].values.reshape(-1,1)) # finding the m
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.va

# Now standardize the data with above maen and variance.
price_standardized_train = project_data_X_train['price'].values#price_scalar.transfo
# Now standardize the data with above maen and variance.
price_standardized_test = project_data_X_test['price'].values#price_scalar.transform
```

```
Mean : 300.16052625, Standard deviation : 355.6141673413906
```

In [103]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn
from sklearn.preprocessing import StandardScaler,normalize

# 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.   .
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data_X_train['teacher_number_of_previously_posted_projects']
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.va

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_train = project_data_X_trai

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_test = project_data_X_test
```

```
Mean : 10.867625, Standard deviation : 27.089926206237163
```

In [113]:
```
wc_title_train=[]
for i in project_data_X_train.project_title.values:
    wc_title_train.append(len(i.split(' ')))
project_data_X_train['wc_title']=wc_title_train

wc_title_test=[]
for i in project_data_X_test.project_title.values:
    wc_title_test.append(len(i.split(' ')))
project_data_X_test['wc_title']=wc_title_test
```

```
wc_title_essay_combined_train=[]
for i in project_data_X_train.combined_essay_title.values:
    wc_title_essay_combined_train.append(len(i.split(' ')))
project_data_X_train['wc_title_essay_combined']=wc_title_essay_combined_train

wc_title_essay_combined_test=[]
for i in project_data_X_test.combined_essay_title.values:
    wc_title_essay_combined_test.append(len(i.split(' ')))
project_data_X_test['wc_title_essay_combined']=wc_title_essay_combined_test
```

In [116]:
```
# 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 mat
self_w2v = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_sta
print(self_w2v.shape)
self_w2v_test= hstack((categories_one_hot_test, sub_categories_one_hot_test,school_s
print(self_w2v_test.shape)
```

```
(8000, 358)
(2000, 358)
```

2.5 Apply XGBoost on the Final Features from the above section
https://xgboost.readthedocs.io/en/latest/python/python_intro.html

In [136]:
```
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_
```

```python
    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],
# }


parameters = {
    'num_boost_round': [10, 20, 30],
    'max_depth': [1, 2, 3, 6, 9]
}


clf = GridSearchCV(clf, parameters,verbose=10,n_jobs=4)
X = self_w2v
Y = project_data_Y_train
```

```
clf.fit(X, Y)
```

```
Fitting 3 folds for each of 15 candidates, totalling 45 fits
```

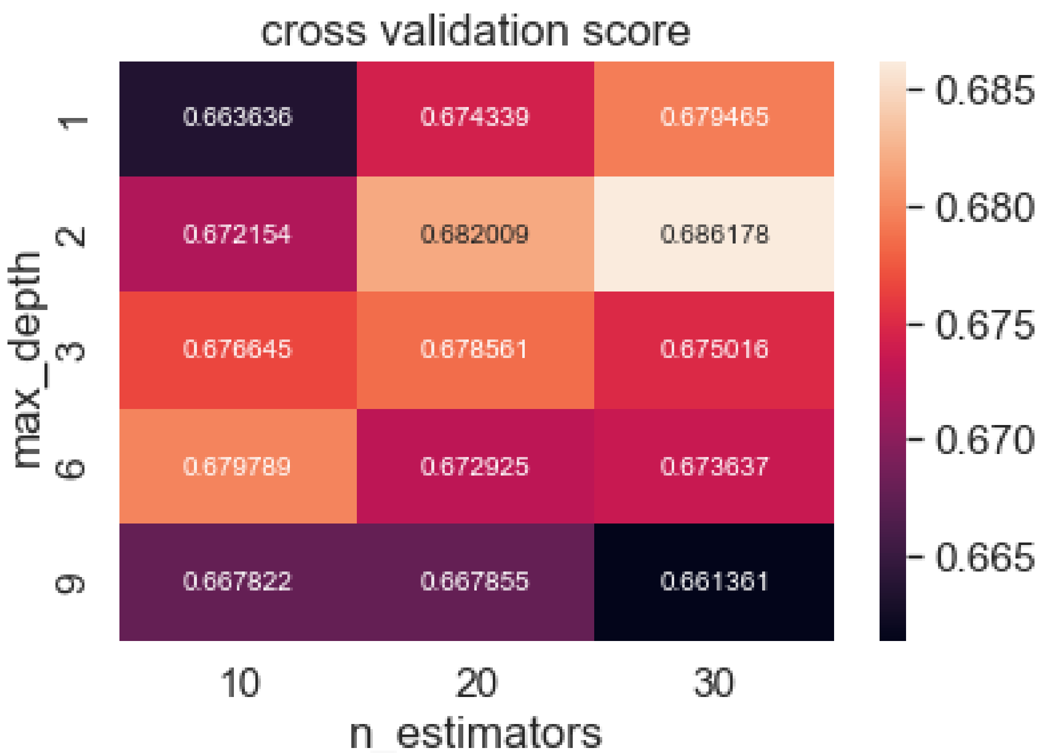
```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   5 tasks      | elapsed:    0.4s
[Parallel(n_jobs=4)]: Done  10 tasks      | elapsed:    0.7s
[Parallel(n_jobs=4)]: Done  17 tasks      | elapsed:    1.4s
[Parallel(n_jobs=4)]: Done  24 tasks      | elapsed:    2.1s
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:    3.7s
[Parallel(n_jobs=4)]: Done  43 out of  45 | elapsed:    6.0s remaining:    0.2s
[Parallel(n_jobs=4)]: Done  45 out of  45 | elapsed:    8.0s finished
```

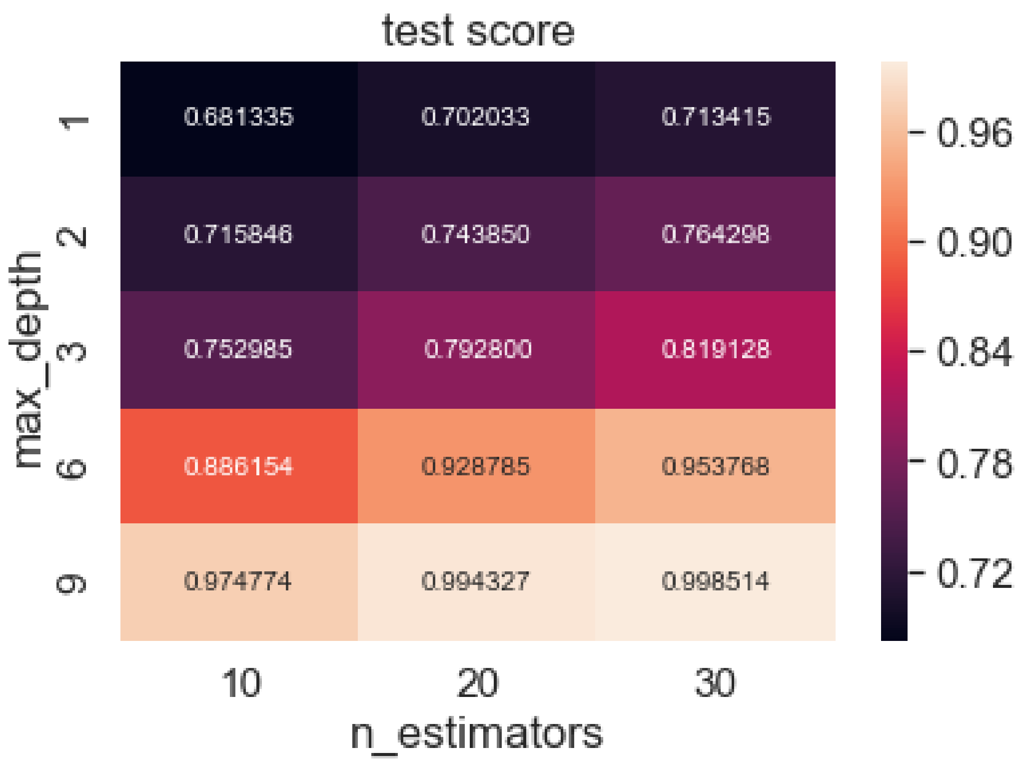
```
Out[136]: GridSearchCV(cv='warn', error_score='raise-deprecating',
                 estimator=<__main__.XGBoostClassifier object at 0x000001FD85075438>,
                 fit_params=None, iid='warn', n_jobs=4,
                 param_grid={'num_boost_round': [10, 20, 30], 'max_depth': [1, 2, 3, 6, 9]},
                 pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                 scoring=None, verbose=10)
```

```
In [137]: #https://stackoverflow.com/questions/30522724/take-multiple-lists-into-dataframe
          #https://seaborn.pydata.org/generated/seaborn.heatmap.html
          max_depth_all=[]
          min_samples_split_all=[]
          for i in range (0,len(clf.cv_results_['params'])):
              max_depth_all.append(clf.cv_results_['params'][i]['max_depth'])
              min_samples_split_all.append(clf.cv_results_['params'][i]['num_boost_round'])
          #print(max_depth_all)
          #print(min_samples_split_all)
          cv_score_all=clf.cv_results_['mean_test_score']
          #print(cv_score_all)
          cv_data=pd.DataFrame(
              {'max_depth': max_depth_all,
               'n_estimators': min_samples_split_all,
               'cv_auc': cv_score_all
              })
          cv_data=cv_data.pivot('max_depth','n_estimators','cv_auc')
          plt.figure(112)
          plt.title("cross validation score")
          sns.heatmap(cv_data, annot=True,annot_kws={"size": 10}, fmt="f")

          train_score_all=clf.cv_results_['mean_train_score']
          #print(train_score_all)
          tain_data=pd.DataFrame(
              {'max_depth': max_depth_all,
               'n_estimators': min_samples_split_all,
               'train_auc': train_score_all
```

```
    })
tain_data=tain_data.pivot('max_depth','n_estimators','train_auc')
plt.figure(122)
plt.title("test score")
sns.heatmap(tain_data, annot=True,annot_kws={"size": 10}, fmt="f")
```

Out[137]: <matplotlib.axes._subplots.AxesSubplot at 0x1fd850777f0>



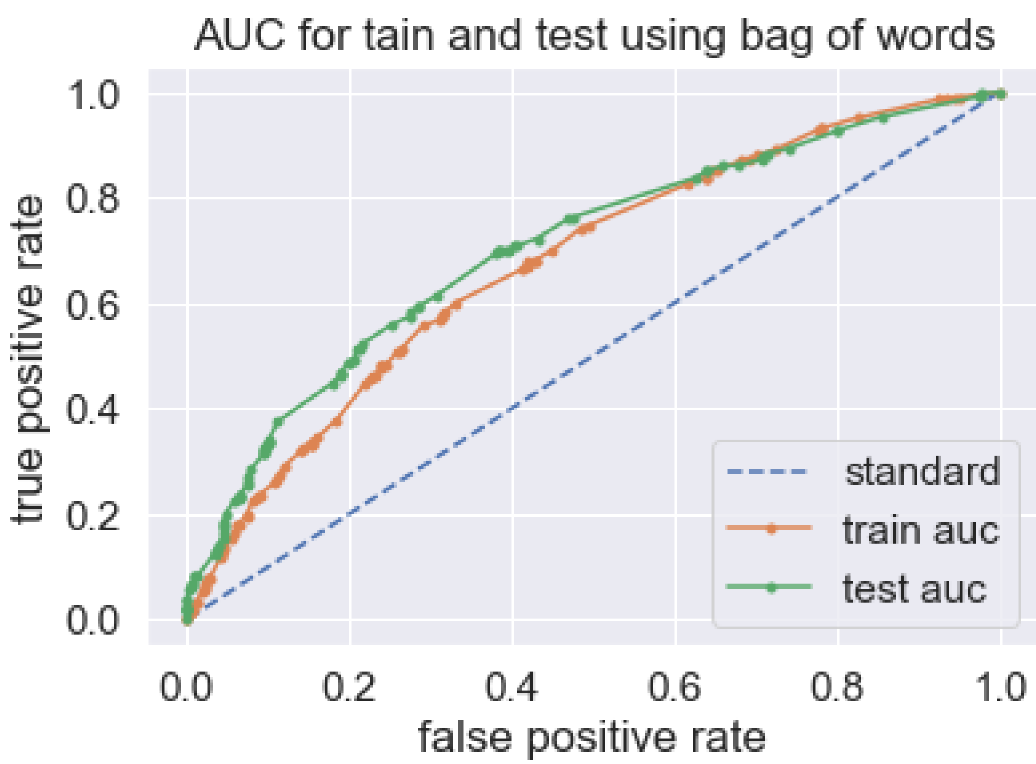



```
In [138]: model=XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4,num_boost_rou
          model.fit(self_w2v,project_data_Y_train)
```

```
In [139]: #https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-class
          from sklearn.metrics import roc_curve
          from sklearn.metrics import roc_auc_score
          from tqdm import tqdm

          probs_test = model.predict_proba(self_w2v_test)
          # keep probabilities for the positive outcome only
          probs_test = probs_test[:, 1]
          auc_test = roc_auc_score(project_data_Y_test, probs_test)
          print('AUC: %.3f' % auc_test)
          fpr, tpr, thresholds = roc_curve(project_data_Y_test, probs_test)

          probs_train = model.predict_proba(self_w2v)
          # keep probabilities for the positive outcome only
          probs_train = probs_train[:, 1]
          auc_train = roc_auc_score(project_data_Y_train, probs_train)
          print('AUC: %.3f' % auc_train)
          fpr1, tpr1, thresholds1 = roc_curve(project_data_Y_train, probs_train)
```

```
plt.plot([0, 1], [0, 1], linestyle='--')
plt.plot(fpr1, tpr1, marker='.')
plt.plot(fpr, tpr, marker='.')

plt.legend({"standard":"","train auc":"","test auc":""})
plt.title("AUC for tain and test using bag of words")
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.show()
```

```
AUC: 0.704
AUC: 0.677
```

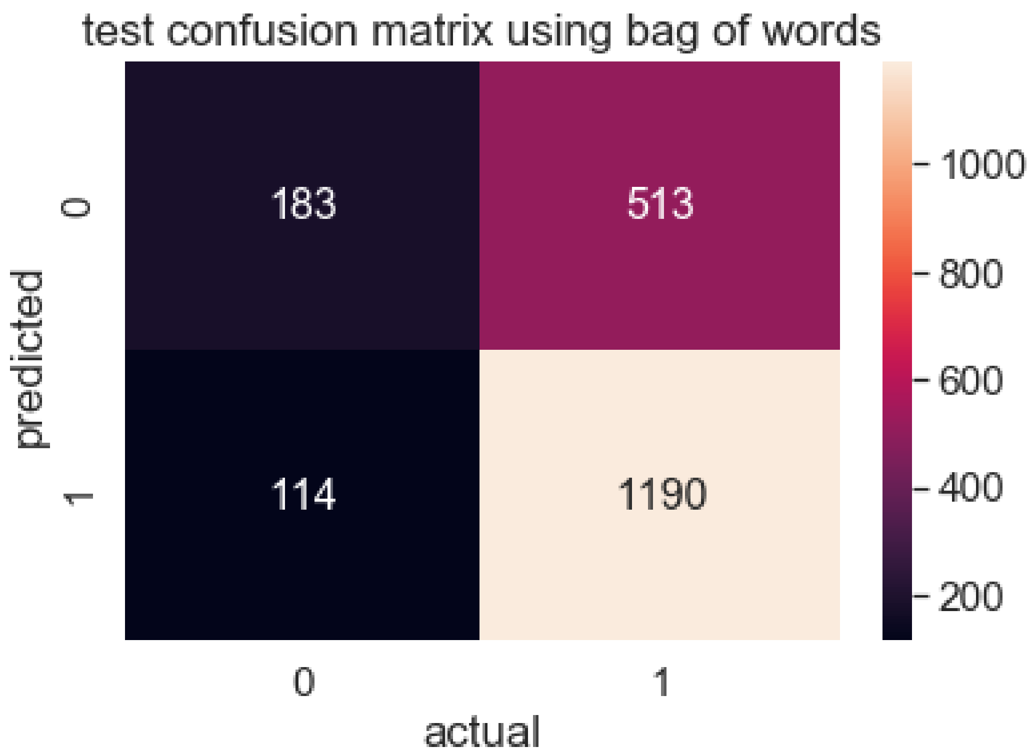


In [140]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    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.r
```

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

In [141]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#compute confudion matrix values and plot
from sklearn.metrics import confusion_matrix
predicted_bow_test=model.predict(self_w2v_test)
tn, fp, fn, tp = confusion_matrix(project_data_Y_test, predict(probs_test, thresholds
print(tn, fp, fn, tp)
print("true positive rate",(tp/(tp+fn)))
print("true negaitive rate",(tn/(tn+fp)))
matrix=[[tn,fn],[fp,tp]]
print(matrix)
df_cm = pd.DataFrame(matrix, range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')# font size
plt.title("test confusion matrix using bag of words")
plt.xlabel("actual")
plt.ylabel("predicted")
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.4305533313166901 for threshold 0.837
183 114 513 1190
true positive rate 0.6987668819729889
true negaitive rate 0.6161616161616161
[[183, 513], [114, 1190]]
```



3. Conclusion

Though the word vector did not work well, its resonably okay for 10k points.

Best parameters for XgboostClassifier are num_boost_round=10 and max_depth=1

The assignment gave clear idea on how to create own word vector and clarified all doubts of matrix decomposition.

```
In [ ]:
```