DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in nee volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoos

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three r

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

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a tea descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then u to need further review before approval.

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One o • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the pro • `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). **Ex**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Examp** • `My students need hands on literacy materials`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*

Feature	Description
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2(
teacher_id	A unique identifier for the teacher of the proposed project. **Exa**
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 sa

* 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. Eac project:

Feature	Description
id	A `project_id` value from the `train.csv` file. **Example:** `p036502`
description	Desciption of the resource. **Example:** `Tenor Saxophone Reeds,`
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 u 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 v

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 e

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about t school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' lea

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 ar

```
!pip install chart-studio
```

```
Requirement already satisfied: chart-studio in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from ch
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-pa
```

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
from tqdm import tqdm
import os
from chart_studio.plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
path_train="/content/drive/My Drive/Colab Notebooks/train_new_data.csv"
```

```
path_resource="/content/drive/My Drive/Assignments_DonorsChoose_2018/resources.csv"
```

1.1 Reading Data

```
project_data_= pd.read_csv(path_train)
resource_data_= pd.read_csv(path_resource)
```

```
project_data=project_data_.head(100000)
resource_data=resource_data_.head(100000)
```

```
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 (100000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_sta
 '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']
```

```
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 (100000, 4)
['id' 'description' 'quantity' 'price']
```

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

1.2 preprocessing of `project_subject_categories`

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/408

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

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

1.3 preprocessing of project_subject_subcategories

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/408

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

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

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
```

```
catogories = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/408

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

project_data['clean_pgc'] = pgc_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
```

1.3 Text preprocessing

```
# 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)
```

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

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	0	p036502	484aaf11257089a66cfedc9461c6bd0a	Ms.	NV
1	3	p185307	525fdbb6ec7f538a48beebaa0a51b24f	Mr.	NC

1.4.2.3 Using Pretrained Models: TFIDF weighted W2V

```
# 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)
```

```
Most of my kindergarten students come from low-income households and are considered \
==================================================
Our school is located the second smallest city in Los Angeles County. Our elementary
==================================================
Our Pre-K students come from very diverse backgrounds. Many come through our doors wi
==================================================
Chicago schools, like many urban school districts across America, have been fighting
==================================================
```

```
# 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


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

```
print("="*50)
```

```
Chicago schools, like many urban school districts across America, have been fighting
==================================================
```

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

```
Chicago schools, like many urban school districts across America, have been fighting
```

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

```
Chicago schools like many urban school districts across America have been fighting ag
```

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

```
# 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=sent.lower()
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 100000/100000 [00:54<00:00, 1847.10it/s]
```

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

```
'chicago schools like many urban school districts across america fighting challenges
```

```
project_data["clean_essays"] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
```

1.4 Preprocessing of `project_title`

```
preprocessed_pt = []
for titles in tqdm(project_data["project_title"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title=title.lower()
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_pt.append(title.lower().strip())
```

```
100%|██████████| 100000/100000 [00:02<00:00, 41136.42it/s]
```

```
project_data["clean_pt"] = preprocessed_pt
project_data.drop(['project_title'], axis=1, inplace=True)
```

```
title_word_count = []
for i in project_data["clean_pt"] :
    j = len(i.split())
    title_word_count.append(j)
project_data["title_word_count"] = title_word_count
project_data.head(5)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	0	p036502	484aaf11257089a66cfedc9461c6bd0a	Ms.	NV
1	3	p185307	525fdbb6ec7f538a48beebaa0a51b24f	Mr.	NC
2	4	p013780	a63b5547a7239eae4c1872670848e61a	Mr.	CA
3	5	p063374	403c6783e9286e51ab318fba40f8d729	Mrs.	DE
4	6	p103285	4e156c5fb3eea2531601c8736f3751a7	Mrs.	MC

```
essay_word_count = []
for i in project_data["clean_essays"] :
    j = len(i.split())
    essay_word_count.append(j)
project_data["essay_word_count"] = essay_word_count
project_data.head(5)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	0	p036502	484aaf11257089a66cfedc9461c6bd0a	Ms.	NV
1	3	p185307	525fdbb6ec7f538a48beebaa0a51b24f	Mr.	NC
2	4	p013780	a63b5547a7239eae4c1872670848e61a	Mr.	CA
3	5	p063374	403c6783e9286e51ab318fba40f8d729	Mrs.	DE
4	6	p103285	4e156c5fb3eea2531601c8736f3751a7	Mrs.	MC

```
project_data['text']=project_data["clean_essays"].map(str) +\
                        project_data["clean_pt"].map(str)
```

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

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

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

analyser = SentimentIntensityAnalyzer()
```

```
neg = []
pos = []
neu = []
compound = []
for i in tqdm(project_data["clean_essays"]) :
    j = analyser.polarity_scores(i)['neg']
    k = analyser.polarity_scores(i)['pos']
    l = analyser.polarity_scores(i)['neu']
    m = analyser.polarity_scores(i)['compound']
    neg.append(j)
    pos.append(k)
    neu.append(l)
    compound.append(m)
```

```
100%|██████████| 100000/100000 [12:03<00:00, 138.23it/s]
```

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

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

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	0	p036502	484aaf11257089a66cfedc9461c6bd0a	Ms.	NV
1	3	p185307	525fdbb6ec7f538a48beebaa0a51b24f	Mr.	NC

```
from sklearn.model_selection import train_test_split
S_train, S_test, y_train, y_test = train_test_split(project_data,
project_data['project_is_approved'], test_size=0.33, stratify = project_data['project_is_approved
])
S_train, S_cv, y_train, y_cv = train_test_split(S_train, y_train, test_size=0.30, stratify=y_trai
```

```
S_train.drop(['project_is_approved'], axis=1, inplace=True)
S_test.drop(['project_is_approved'], axis=1, inplace=True)
S_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

1.5 Preparing data for models

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'clean_pgc', 'clean_essays',
       'clean_pt', 'title_word_count', 'essay_word_count', 'text', 'neg',
       'pos', 'neu', 'compound'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data

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

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

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

```
my_counter = Counter()
for word in S_train['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]))
```

```
from collections import Counter
my_counter = Counter()
for word in S_train['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]))
```

```
from collections import Counter
my_counter = Counter()
for word in S_train['school_state'].values:
    my_counter.update(word.split())
state_dict = dict(my_counter)
sorted_state_dict = dict(sorted(state_dict.items(), key=lambda kv: kv[1]))
```

```
my_counter = Counter()
for word in S_train['teacher_prefix'].values:
    my_counter.update(word.split())
prefix_dict = dict(my_counter)
sorted_prefix_dict = dict(sorted(prefix_dict.items(), key=lambda kv: kv[1]))
```

```
from collections import Counter
my_counter = Counter()
for word in S_train['clean_pgc'].values:
    my_counter.update(word.split())

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

1.5.1 Vectorizing Categorical data

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

one hot encoding for clean categories

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_clean_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
vectorizer_clean_cat.fit(S_train['clean_categories'].values)
categories_one_hot_train = vectorizer_clean_cat.transform(S_train['clean_categories'].values)
categories_one_hot_test = vectorizer_clean_cat.transform(S_test['clean_categories'].values)
categories_one_hot_cv = vectorizer_clean_cat.transform(S_cv['clean_categories'].values)
print(vectorizer_clean_cat.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",categories_one_hot_test.shape)
print("Shape of matrix of CV data after one hot encoding ",categories_one_hot_cv.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'Special
Shape of matrix of Train data after one hot encoding  (46900, 9)
Shape of matrix of Test data after one hot encoding  (33000, 9)
Shape of matrix of CV data after one hot encoding  (20100, 9)
```

one hot encoding for subcategories

```
vectorizer_clean_subcat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=
True)
vectorizer_clean_subcat.fit(S_train['clean_subcategories'].values)
sub_categories_one_hot_train = vectorizer_clean_subcat.transform(S_train['clean_subcategories'].v
sub_categories_one_hot_test = vectorizer_clean_subcat.transform(S_test['clean_subcategories'].val
sub_categories_one_hot_cv = vectorizer_clean_subcat.transform(S_cv['clean_subcategories'].values)
print(vectorizer_clean_subcat.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",sub_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",sub_categories_one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",sub_categories_one_hot_c
.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracur
Shape of matrix of Train data after one hot encoding  (46900, 30)
Shape of matrix of Test data after one hot encoding  (33000, 30)
Shape of matrix of Cross Validation data after one hot encoding  (20100, 30)
```

one hot encoding for state

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
vectorizer_school_state= CountVectorizer(vocabulary=list(sorted_state_dict.keys()), lowercase=Fal
True)
vectorizer_school_state.fit(S_train['school_state'].values)
school_state_one_hot_train = vectorizer_school_state.transform(S_train['school_state'].values)
school_state_one_hot_test = vectorizer_school_state.transform(S_test['school_state'].values)
school_state_one_hot_cv = vectorizer_school_state.transform(S_cv['school_state'].values)
print(vectorizer_school_state.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",school_state_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",school_state_one_hot_cv
.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'AK', 'NH', 'DE', 'DC', 'HI', 'ME', 'WV',
Shape of matrix of Train data after one hot encoding  (46900, 51)
Shape of matrix of Test data after one hot encoding  (33000, 51)
Shape of matrix of Cross Validation data after one hot encoding  (20100, 51)
```

one hot encoding for teacher prefix

```
vectorizer_prefix = CountVectorizer(vocabulary=list(sorted_prefix_dict.keys()), lowercase=False,
True)
vectorizer_prefix.fit(S_train['teacher_prefix'].values)
teacher_prefix_one_hot_train = vectorizer_prefix.transform(S_train['teacher_prefix'].values)
teacher_prefix_one_hot_test = vectorizer_prefix.transform(S_test['teacher_prefix'].values)
teacher_prefix_one_hot_cv = vectorizer_prefix.transform(S_cv['teacher_prefix'].values)
print(vectorizer_prefix.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",teacher_prefix_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",teacher_prefix_one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",teacher_prefix_one_hot_c
.shape)
```

```
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix of Train data after one hot encoding  (46900, 5)
Shape of matrix of Test data after one hot encoding  (33000, 5)
Shape of matrix of Cross Validation data after one hot encoding  (20100, 5)
```

one hot encoding for project grade category

```
vectorizer_pgc= CountVectorizer(vocabulary=list(sorted_pgc_dict.keys()), lowercase=False, binary=
True)
vectorizer_pgc.fit(S_train['clean_pgc'].values)
clean_project_grade_category_one_hot_train = vectorizer_pgc.transform(S_train['clean_pgc'].values
clean_project_grade_category_one_hot_test = vectorizer_pgc.transform(S_test['clean_pgc'].values)
clean_project_grade_category_one_hot_cv = vectorizer_pgc.transform(S_cv['clean_pgc'].values)
print(vectorizer_pgc.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",clean_project_grade_category_one_ho
print("Shape of matrix of Test data after one hot encoding ",clean_project_grade_category_one_hot
print("Shape of matrix of Cross Validation data after one hot encoding ",clean_project_grade_cate
.shape)
```

```
['Grades9-12', 'Grades6-8', 'Grades3-5', 'GradesPreK-2']
Shape of matrix of Train data after one hot encoding  (46900, 4)
Shape of matrix of Test data after one hot encoding  (33000, 4)
Shape of matrix of Cross Validation data after one hot encoding  (20100, 4)
```

1.5.2.2 TFIDF vectorizer

```
text_data=project_data['text']
text_data.shape
```

```
(100000,)
```

1.5.3 Vectorizing Numerical features

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

```
S_train = pd.merge(S_train, price_data, on='id', how='left')
S_test = pd.merge(S_test, price_data, on='id', how='left')
S_cv = pd.merge(S_cv, price_data, on='id', how='left')
```

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

```
price_scalar = Normalizer()
X=S_train['price'].fillna(S_train['price'].mean())
Y=S_test['price'].fillna(S_test['price'].mean())
Z=S_cv['price'].fillna(S_cv['price'].mean())
price_scalar.fit(X.values.reshape(1,-1)) # finding the mean and standard deviation of this data
price_standardized_train = price_scalar.transform(X.values.reshape(1, -1))
price_standardized_test = price_scalar.transform(Y.values.reshape(1, -1))
price_standardized_cv = price_scalar.transform(Z.values.reshape(1, -1))
```

```
print(price_standardized_train.shape)
print(price_standardized_test.shape)
print(price_standardized_cv.shape)
```

```
(1, 46900)
(1, 33000)
(1, 20100)
```

Normalizing teacher posted projects

```
price_scalar.fit(S_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
prev_project_standardized_train = price_scalar.transform(S_train['teacher_number_of_previously_po
prev_project_standardized_test = price_scalar.transform(S_test['teacher_number_of_previously_post
prev_project_standardized_cv = price_scalar.transform(S_cv['teacher_number_of_previously_posted_p
```

```
print(prev_project_standardized_train.shape)
print(prev_project_standardized_test.shape)
print(prev_project_standardized_cv.shape)
```

```
(1, 46900)
(1, 33000)
(1, 20100)
```

Normalizing quantity

```
X=S_train['quantity'].fillna(S_train['quantity'].mean())
Y=S_test['quantity'].fillna(S_test['quantity'].mean())
Z=S_cv['quantity'].fillna(S_cv['quantity'].mean())
price_scalar.fit(X.values.reshape(1,-1)) # finding the mean and standard deviation of this data
quantity_standardized_train = price_scalar.transform(X.values.reshape(1, -1))
quantity_standardized_test = price_scalar.transform(Y.values.reshape(1, -1))
quantity_standardized_cv = price_scalar.transform(Z.values.reshape(1, -1))
```

```
print(quantity_standardized_train.shape)
print(quantity_standardized_test.shape)
print(quantity_standardized_cv.shape)
```

```
(1, 46900)
(1, 33000)
(1, 20100)
```

Normalizing title word count

```
normalizer = Normalizer()
normalizer.fit(S_train['title_word_count'].values.reshape(1,-1))
title_word_count_train = normalizer.transform(S_train['title_word_count'].values.reshape(1,-1))
title_word_count_cv = normalizer.transform(S_cv['title_word_count'].values.reshape(1,-1))
```

```
title_word_count_test = normalizer.transform(S_test['title_word_count'].values.reshape(1,-1))
print("After vectorizations")
print(title_word_count_train.shape, y_train.shape)
print(title_word_count_cv.shape, y_cv.shape)
print(title_word_count_test.shape, y_test.shape)
```

```
After vectorizations
(1, 46900) (46900,)
(1, 20100) (20100,)
(1, 33000) (33000,)
```

Normalizing essay word counts

```
normalizer = Normalizer()
normalizer.fit(S_train['essay_word_count'].values.reshape(1,-1))
essay_word_count_train = normalizer.transform(S_train['essay_word_count'].values.reshape(1,-1))
essay_word_count_cv = normalizer.transform(S_cv['essay_word_count'].values.reshape(1,-1))
essay_word_count_test = normalizer.transform(S_test['essay_word_count'].values.reshape(1,-1))
print("After vectorizations")
print(essay_word_count_train.shape, y_train.shape)
print(essay_word_count_cv.shape, y_cv.shape)
print(essay_word_count_test.shape, y_test.shape)
```

```
After vectorizations
(1, 46900) (46900,)
(1, 20100) (20100,)
(1, 33000) (33000,)
```

Normalizing positive sentiment

```
normalizer = Normalizer()
normalizer.fit(S_train['pos'].values.reshape(1,-1))
essay_sent_pos_train = normalizer.transform(S_train['pos'].values.reshape(1,-1))
essay_sent_pos_cv = normalizer.transform(S_cv['pos'].values.reshape(1,-1))
essay_sent_pos_test = normalizer.transform(S_test['pos'].values.reshape(1,-1))
print("After vectorizations")
print(essay_sent_pos_train.shape, y_train.shape)
print(essay_sent_pos_cv.shape, y_cv.shape)
print(essay_sent_pos_test.shape, y_test.shape)
```

```
After vectorizations
(1, 46900) (46900,)
(1, 20100) (20100,)
(1, 33000) (33000,)
```

Normalizing negative sentiment

```
normalizer = Normalizer()
normalizer.fit(S_train['neg'].values.reshape(1,-1))
essay_sent_neg_train = normalizer.transform(S_train['neg'].values.reshape(1,-1))
essay_sent_neg_cv = normalizer.transform(S_cv['neg'].values.reshape(1,-1))
essay_sent_neg_test = normalizer.transform(S_test['neg'].values.reshape(1,-1))
print("After vectorizations")
print(essay_sent_neg_train.shape, y_train.shape)
print(essay_sent_neg_cv.shape, y_cv.shape)
print(essay_sent_neg_test.shape, y_test.shape)
```

```
After vectorizations
(1, 46900) (46900,)
(1, 20100) (20100,)
(1, 33000) (33000,)
```

Normalizing neutral sentiment

```
normalizer = Normalizer()
normalizer.fit(S_train['neu'].values.reshape(1,-1))
essay_sent_neu_train = normalizer.transform(S_train['neu'].values.reshape(1,-1))
essay_sent_neu_cv = normalizer.transform(S_cv['neu'].values.reshape(1,-1))
essay_sent_neu_test = normalizer.transform(S_test['neu'].values.reshape(1,-1))
print("After vectorizations")
print(essay_sent_neu_train.shape, y_train.shape)
print(essay_sent_neu_cv.shape, y_cv.shape)
print(essay_sent_neu_test.shape, y_test.shape)
```

```
After vectorizations
(1, 46900) (46900,)
(1, 20100) (20100,)
(1, 33000) (33000,)
```

Normalizing compound sentiment

```
normalizer = Normalizer()
normalizer.fit(S_train['compound'].values.reshape(1,-1))
essay_sent_comp_train = normalizer.transform(S_train['compound'].values.reshape(1,-1))
essay_sent_comp_cv = normalizer.transform(S_cv['compound'].values.reshape(1,-1))
essay_sent_comp_test = normalizer.transform(S_test['compound'].values.reshape(1,-1))
print("After vectorizations")
print(essay_sent_comp_train.shape, y_train.shape)
print(essay_sent_comp_cv.shape, y_cv.shape)
print(essay_sent_comp_test.shape,y_test.shape)
```

```
After vectorizations
(1, 46900) (46900,)
(1, 20100) (20100,)
(1, 33000) (33000,)
```

Assignment 11: TruncatedSVD

- step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title a 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 numl method

    - The shape of the matrix after TruncatedSVD will be 2000*n, i.e. each row represents a vector f
    - Vectorize the essay text and project titles using these word vectors. (while vectorizing, do ignc 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: XGBO**
- **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 h**

2. TruncatedSVD

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

```
tfidf_vector= TfidfVectorizer()
data_tf = tfidf_vector.fit_transform(text_data)
```

```
print(tfidf_vector)
```

```
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)
```

```
sorted_features = np.argsort(tfidf_vector.idf_)
features = tfidf_vector.get_feature_names()
top_features = [features[i] for i in sorted_features[:2000]]
```

```
len(top_features)
```

```
2000
```

```
top_features
```

```
['students',
 'school',
 'learning',
 'classroom',
 'not',
 'learn',
 'help',
 'many',
 'need',
 'work',
 'come',
 'use',
 'able',
 'love',
 'day',
 'also',
 'class',
 'make',
 'new',
 'year',
 'one',
 'would',
 'time',
 'student',
 'want',
 'skills',
 'grade',
 'reading',
 'get',
 'every',
 'allow',
 'provide',
 'teach',
 'free',
 'high',
 'like',
 'project',
 'way',
 'technology',
 'different',
 'learners',
 'world',
 'materials',
 'best',
 'lunch',
 'well',
 'group',
 'children',
 'needs',
 'education',
 'create',
 'give',
 'life',
 'home',
 'teacher',
 'math',
 'read',
 'using',
 'opportunity',
 'hard',
 'working',
```

```
        'low',
        'community',
        'books',
        'eager',
        'first',
        'fun',
        'take',
        'excited',
        'activities',
        'see',
        'title',
        'small',
        'kids',
        'great',
        'environment',
        'language',
        'access',
        'us',
        'diverse',
        'much',
        'hands',
        'become',
        'know',
        'used',
        'daily',
        'experience',
        'level',
        'better',
        'even',
        'around',
        'resources',
        'things',
        'income',
        'poverty',
        'special',
        'support',
        'english',
        'families',
        'backgrounds',
        'receive',
        'graders',
        'always',
        'successful',
        'variety',
        'science',
        'keep',
        'reduced',
        'important',
        'place',
        'opportunities',
        'supplies',
        'ready',
        'focus',
        'area',
        'experiences',
        'challenges',
        'together',
        'feel',
        'goal',
        'academic',
        'ways',
        'part'
```

```
 'part',
 'move',
 'writing',
 'teaching',
 'often',
 'go',
 'creative',
 'future',
 'amazing',
 'years',
 'practice',
 'projects',
 'lives',
 'enjoy',
 'order',
 'program',
 'no',
 'success',
 'throughout',
 'full',
 'build',
 'explore',
 'teachers',
 'tools',
 'room',
 'play',
 'district',
 'grow',
 'without',
 'book',
 'engaged',
 'social',
 'little',
 'kindergarten',
 'live',
 'find',
 'research',
 'may',
 'however',
 'set',
 'parents',
 'knowledge',
 'others',
 'improve',
 'face',
 'safe',
 'active',
 'share',
 'meet',
 'second',
 'engaging',
 'needed',
 'child',
 'lessons',
 'everyday',
 'outside',
 'ability',
 'two',
 'groups',
 'still',
 'young',
 'despite',
```

```
        'building',
        'could',
        'population',
        'based',
        'seating',
        'develop',
        'instruction',
        'wonderful',
        'limited',
        'levels',
        'sit',
        'readers',
        'try',
        'currently',
        'increase',
        'making',
        'continue',
        'long',
        'bring',
        'believe',
        'various',
        'engage',
        'positive',
        'show',
        'possible',
        'benefit',
        'family',
        'lot',
        'difficult',
        'games',
        'succeed',
        'another',
        'something',
        'curriculum',
        'literacy',
        'educational',
        'good',
        'classes',
        'providing',
        'problem',
        'addition',
        'creating',
        'large',
        'several',
        'real',
        'areas',
        'unique',
        'breakfast',
        'struggle',
        'available',
        'library',
        'city',
        'elementary',
        'really',
        'enough',
        'thinking',
        'hope',
        'comfortable',
        'goals',
        'never',
        'computer',
        'challenge',
```

```
 'challenge',
 'complete',
 'energy',
 'items',
 'allows',
 'times',
 'understand',
 'homes',
 'right',
 'basic',
 'art',
 'exciting',
 'chairs',
 'look',
 'majority',
 'concepts',
 'choose',
 'think',
 'makes',
 'middle',
 'center',
 'given',
 'stay',
 'write',
 'reach',
 'space',
 'start',
 'old',
 'enhance',
 'understanding',
 'within',
 'getting',
 'due',
 'located',
 'individual',
 'encourage',
 'looking',
 'chance',
 'movement',
 'means',
 'potential',
 'range',
 'please',
 'whole',
 'words',
 'helping',
 'coming',
 'deserve',
 'strive',
 'paper',
 'helps',
 'age',
 'self',
 'schools',
 'rural',
 'motivated',
 'interest',
 'last',
 'going',
 'energetic',
 'necessary',
 'sitting',
```

```
        'put',
        'independent',
        'activity',
        'minds',
        'lack',
        'requesting',
        'everything',
        'made',
        'ideas',
        'peers',
        'people',
        'change',
        'next',
        'achieve',
        'difference',
        'economic',
        'information',
        'extra',
        'physical',
        'classrooms',
        'studies',
        'stories',
        'attend',
        'back',
        'wide',
        'thank',
        'since',
        'participate',
        'socioeconomic',
        'especially',
        'walk',
        'truly',
        'arts',
        'moving',
        'options',
        'problems',
        'basis',
        'focused',
        'giving',
        'computers',
        'flexible',
        'process',
        'attention',
        'college',
        'growth',
        'disabilities',
        '100',
        'critical',
        'abilities',
        'grades',
        'creativity',
        'programs',
        'prepare',
        'along',
        'donation',
        'curious',
        'third',
        'centers',
        'big',
        'neighborhood',
        'yet',
        'end'.
```

```
 'setting',
 'century',
 'ever',
 'multiple',
 'sure',
 'interactive',
 'current',
 'today',
 'stem',
 'asking',
 'job',
 'meaningful',
 'growing',
 'public',
 'trying',
 'everyone',
 'beyond',
 'already',
 'including',
 'single',
 'personal',
 'state',
 'core',
 'chromebooks',
 'ipads',
 'let',
 'sometimes',
 'inspire',
 'quality',
 'ipad',
 'urban',
 'donations',
 '21st',
 'open',
 'cannot',
 'week',
 'learned',
 'add',
 'standards',
 'playing',
 'taught',
 'common',
 'spend',
 'team',
 'desire',
 'ensure',
 'three',
 'essential',
 'choice',
 'challenging',
 'number',
 'strong',
 'music',
 'must',
 'bright',
 'greatly',
 'equipment',
 'begin',
 'healthy',
 'comes',
 'allowing',
```

```
'care',
'ask',
'taking',
'offer',
'tool',
'hand',
'forward',
'unfortunately',
'extremely',
'constantly',
'task',
'provides',
'excitement',
'speak',
'funding',
'solving',
'early',
'living',
'online',
'thing',
'confidence',
'requested',
'questions',
'parent',
'topics',
'impact',
'desks',
'sense',
'solve',
'5th',
'content',
'visual',
'though',
'food',
'comprehension',
'struggling',
'skill',
'listening',
'plan',
'morning',
'apps',
'lesson',
'chair',
'listen',
'enthusiastic',
'balls',
'vocabulary',
'although',
'independently',
'include',
'kind',
'beginning',
'engagement',
'foster',
'past',
'choices',
'gain',
'anything',
'story',
'hear',
'table',
'text',
```

```
 'additional',
 'serve',
 'imagine',
 'qualify',
 'higher',
 'huge',
 'away',
 'faced',
 'easy',
 'emotional',
 'traditional',
 'literature',
 'easily',
 'instead',
 'desk',
 'works',
 'simple',
 'body',
 'days',
 'academically',
 'engineering',
 'money',
 'background',
 'stools',
 'enter',
 'study',
 'less',
 'specific',
 'non',
 'asked',
 'appropriate',
 'four',
 'becoming',
 'turn',
 'word',
 'career',
 'wobble',
 'organized',
 'development',
 'require',
 'autism',
 'entire',
 'close',
 'five',
 'issues',
 'subject',
 'floor',
 'minutes',
 'pre',
 'type',
 'design',
 'american',
 'proud',
 'staff',
 'promote',
 'provided',
 'price',
 'seats',
 'cultures',
 'languages',
 'types',
```

```
 'material',
 'bodies',
 'styles',
 'filled',
 'digital',
 'assignments',
 'therefore',
 'incorporate',
 'brain',
 'budget',
 'found',
 'happy',
 'board',
 'amount',
 'fifth',
 'favorite',
 'key',
 'receiving',
 'behavior',
 'loving',
 'caring',
 'health',
 'matter',
 'purchase',
 'pencils',
 'communication',
 'utilize',
 'express',
 'seen',
 'passion',
 'status',
 'control',
 'collaborate',
 'thrive',
 'achievement',
 'system',
 'interested',
 'subjects',
 'alternative',
 'fluency',
 'academics',
 'developing',
 'fourth',
 'gives',
 'store',
 'expand',
 'game',
 '4th',
 'typical',
 'involved',
 'regular',
 'collaboration',
 'highly',
 'inner',
 'heart',
 'faces',
 'progress',
 'say',
 'enable',
 'inquisitive',
 'tell',
 'awesome',
```

```
'3rd',
'spanish',
'color',
'strategies',
'lots',
'pride',
'friends',
'test',
'tablets',
'towards',
'serves',
'members',
'inspired',
'exposed',
'resource',
'wait',
'percent',
'joy',
'town',
'lab',
'almost',
'pictures',
'across',
'foundation',
'interests',
'etc',
'videos',
'leaders',
'general',
'wiggle',
'headphones',
'perfect',
'afford',
'assist',
'stand',
'fit',
'field',
'interesting',
'exercise',
'situations',
'culture',
'tasks',
'discover',
'created',
'20',
'far',
'individuals',
'done',
'10',
'internet',
'half',
'interact',
'mind',
'history',
'hold',
'willing',
'exposure',
'least',
'door',
'places',
'connect',
'expectations'
```

```
 'expectations',
'sensory',
'motivation',
'households',
'sharing',
'2nd',
'print',
'easier',
'idea',
'rich',
'markers',
'adults',
'step',
'pencil',
'seeing',
'prepared',
'said',
'remember',
'motivate',
'texts',
'donating',
'thinkers',
'seat',
'ball',
'run',
'smart',
'events',
'watch',
'started',
'takes',
'simply',
'longer',
'manipulatives',
'inside',
'tables',
'6th',
'quickly',
'true',
'rather',
'lead',
'motor',
'country',
'obstacles',
'curiosity',
'recess',
'importance',
'60',
'fully',
'stations',
'hispanic',
'recently',
'unit',
'cultural',
'society',
'appreciate',
'present',
'loved',
'letters',
'leave',
'break',
'diversity',
'responsibility',
```

```
        'beautiful',
        'includes',
        'olds',
        'economically',
        'exploring',
        'keeping',
        'behind',
        'ones',
        '8th',
        'productive',
        'gifted',
        'required',
        'hours',
        'balance',
        'model',
        'google',
        'actively',
        'dedicated',
        'sets',
        'overcome',
        'socio',
        'option',
        'lifelong',
        'light',
        'generous',
        'boards',
        'connections',
        'effective',
        '30',
        'might',
        'speaking',
        'writers',
        'changing',
        'apply',
        '90',
        'funds',
        'power',
        'smile',
        'mathematics',
        'communicate',
        'meeting',
        'sports',
        'innovative',
        'station',
        'fact',
        'six',
        'happen',
        'scientists',
        'course',
        'challenged',
        'nannanhelp',
        'scholars',
        'rest',
        'supply',
        'girls',
        'form',
        'carpet',
        'example',
        'devices',
        'african',
        'dreams',
        'dream'
```

```
 'dream',
 'whether',
 'classmates',
 'short',
 'shows',
 'perform',
 'proper',
 'rate',
 'responsible',
 'enthusiasm',
 'boys',
 'lunches',
 'funded',
 'adding',
 'fine',
 'rigorous',
 'jobs',
 'coding',
 'lower',
 'approximately',
 'hardships',
 'numbers',
 'sounds',
 'journey',
 'printer',
 'collaborative',
 'consists',
 'value',
 'met',
 'unable',
 'disadvantaged',
 'struggles',
 '12',
 'possibilities',
 'push',
 'track',
 'services',
 'shown',
 'reader',
 'video',
 'focusing',
 'fall',
 'gap',
 'follow',
 'discuss',
 'white',
 'careers',
 'benefits',
 'top',
 '25',
 'lucky',
 'donors',
 'hoping',
 'generation',
 'finding',
 'citizens',
 'rug',
 'risk',
 'overall',
 'nothing',
 'fiction',
 'starting',
```

```
        'ownership',
        'steam',
        'countries',
        'greater',
        'accomplish',
        'highest',
        'exploration',
        'designed',
        'either',
        'fortunate',
        'talk',
        'performance',
        'endless',
        'eyes',
        'approach',
        'stop',
        'confident',
        'sound',
        'regardless',
        'valuable',
        'incredible',
        'local',
        'path',
        'pieces',
        'hungry',
        'kits',
        'performing',
        'talented',
        'data',
        'display',
        'among',
        'guided',
        'minute',
        'south',
        'actually',
        'knowing',
        'imagination',
        '50',
        'reinforce',
        'accessible',
        'brains',
        'box',
        'considered',
        'standing',
        'storage',
        'feeling',
        'kid',
        'written',
        'picture',
        'realize',
        'campus',
        'financial',
        'media',
        'master',
        'supportive',
        'stability',
        'style',
        'effort',
        'kiddos',
        'shared',
        'mini',
        '1st'
```

```
 'ist',
'testing',
'busy',
'water',
'discussions',
'role',
'adhd',
'preschool',
'presentations',
'breaks',
'consider',
'mostly',
'inviting',
'reality',
'parts',
'view',
'introduce',
'expected',
'smiles',
'crucial',
'letter',
'finally',
'result',
'individualized',
'excel',
'drive',
'supports',
'lost',
'increasing',
'varying',
'instructional',
'colorful',
'completing',
'beneficial',
'fitness',
'answer',
'character',
'differences',
'raised',
'differentiated',
'line',
'maintain',
'gets',
'technological',
'front',
'hardworking',
'county',
'bunch',
'dance',
'ahead',
'requires',
'primary',
'record',
'practicing',
'weekly',
'demonstrate',
'yoga',
'excellence',
'supporting',
'organization',
'states',
'allowed',
```

```
        'charter',
        'clean',
        'vital',
        'effectively',
        'intervention',
        'dry',
        'related',
        'passionate',
        'capable',
        'nannanflexible',
        'expose',
        'strengthen',
        'sight',
```

2.2 Computing Co-occurance matrix

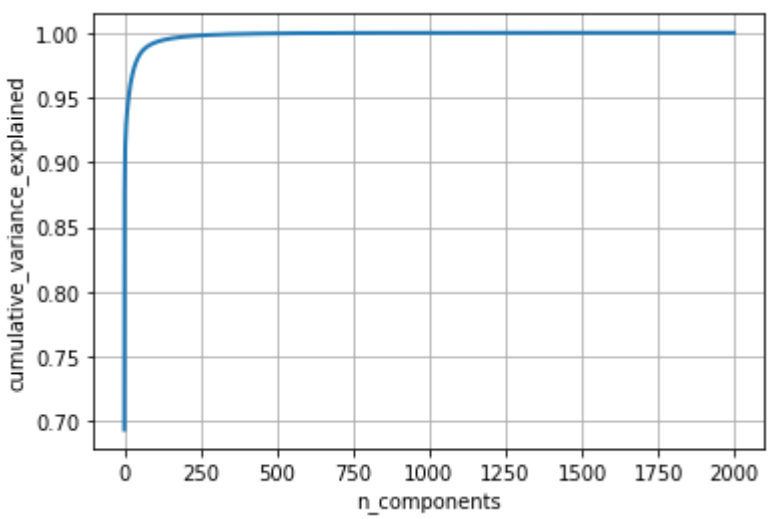
```
length=2000
window=5
m = np.zeros([length,length])
for sentence in tqdm(text_data):
  sent=sentence.split()
  for i,word in enumerate(sent):
    if word in top_features:
      for j in range(max(i-window,0),min(i+window,len(sent)-1)+1):
        if sent[j] in top_features:
          if sent[j]!=word:
            m[top_features.index(word),top_features.index(sent[j])]+=1
          else:
            pass
        else:
          pass
    else:
      pass
print(m)
```

```
100%|██████████| 100000/100000 [39:46<00:00, 41.77it/s][[0.00000e+00 1.29237e+05 9.52
 2.49000e+02]
[1.29237e+05 0.00000e+00 1.80080e+04 ... 8.80000e+01 1.20000e+01
 5.30000e+01]
[9.52050e+04 1.80080e+04 0.00000e+00 ... 1.00000e+02 7.50000e+01
 1.18000e+02]
...
[1.96000e+02 8.80000e+01 1.00000e+02 ... 0.00000e+00 1.00000e+00
 0.00000e+00]
[2.64000e+02 1.20000e+01 7.50000e+01 ... 1.00000e+00 0.00000e+00
 1.00000e+00]
[2.49000e+02 5.30000e+01 1.18000e+02 ... 0.00000e+00 1.00000e+00
 0.00000e+00]]
```

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

```
from sklearn.decomposition import TruncatedSVD
from scipy.sparse import csr_matrix
X_sparse = csr_matrix(m)
tsvd = TruncatedSVD(n_components=X_sparse.shape[1]-1)
X_tsvd = tsvd.fit(m)
percentage_var_explained=tsvd.explained_variance_/np.sum(tsvd.explained_variance_)
cum_var_explained=np.cumsum(percentage_var_explained)
```

```
plt.figure(1,figsize=(6,4))
plt.clf()
plt.plot(cum_var_explained,linewidth=2)
plt.axis('tight')
plt.grid()
plt.xlabel('n_components')
plt.ylabel('cumulative_variance_explained')
plt.show()
```



we can see 100% variance can be saved within 150 dimension

```
svd = TruncatedSVD(n_components = 150)
new_svd_2000 = svd.fit_transform(m)
```

```
new_svd_2000.shape
```

```
(2000, 150)
```

```
print(new_svd_2000[1])
```

```
[ 1.41508313e+05 -6.33448944e+04  3.20492860e+04  4.90220710e+04
 -4.08269613e+03  2.17498263e+03  7.19391628e+02 -1.30168962e+02
 -1.63885602e+03 -4.28840554e+03 -1.70543327e+03 -2.14956286e+03
 -1.17307038e+03  9.84465644e+02 -3.27509665e+01  1.03255805e+03
  4.17898085e+02 -1.54660985e+03  8.89448546e+02  1.98678769e+02
 -4.06897088e+02  4.76549889e+02  5.57737393e+02  2.35250166e+02
 -6.43706744e+02 -4.75718844e+02  4.85096309e+02 -2.78016465e+02
 -5.82458401e+02 -3.67638112e+01  2.35094331e+02 -2.47786957e+02
  1.66548064e+02 -2.04048035e+02 -1.15264305e+02  3.96199571e+02
  5.33685324e+02  1.32012305e+02 -1.84634211e+03  1.95591946e+02
  5.95332315e+02 -1.73127193e+01 -1.02850802e+01 -7.57531771e+02
  4.06175984e+02 -5.49200567e+02 -3.05113306e+02 -1.12237726e+02
  4.08633998e+02  1.41242997e+02 -2.83399765e+02 -1.16830779e+02
 -5.67980089e+01  3.25891286e+02  5.41992466e+01 -4.93875493e+01
  9.67298517e+01 -9.52100570e+01  1.83831541e+02  2.18612126e+02
  6.11067518e+01 -5.16649457e+01 -1.50703564e+02 -7.76459186e+00
 -1.76631666e+02 -9.10684031e+01 -5.18692924e+01 -8.92651147e+01
 -3.13154542e+01 -2.63617706e+01 -1.13499922e+02  1.31739212e+02
 -1.21684976e+02  2.39983568e+02  3.74007080e+01  1.42719847e+01
 -1.19540178e+02  2.94426669e+02  5.16663651e+01 -8.16485854e+01
  5.51866289e+01  1.33633521e+01 -7.79218745e+01  2.86449519e+01
  6.46196172e+01  1.02946756e+02 -2.48853806e+02  2.68993051e+01
  3.77800233e+01 -4.60182908e+01  9.90787203e+00  5.94760066e+01
  4.50326712e+01  5.08871572e+01  2.49226243e+01  6.06988149e+01
  1.18518498e+01  3.24666957e+01  1.63332915e+01  5.37408461e+00
 -3.73253371e+01  4.15119541e+01 -3.36239819e+01  4.57338757e+01
  4.11154118e+00 -1.20862670e+01 -1.84352526e+01 -1.18339227e+01
  1.20052592e+02 -8.32792610e-01 -7.67925405e+01  2.74768695e+01
  4.37030171e+01 -1.87851935e+00 -2.57374892e+01  1.78484887e+01
  1.72160859e+01 -4.20370722e+01  6.59974234e+01  1.48964046e+01
 -8.25445828e+01  5.83090739e+01 -1.81972984e+00  1.45028491e+00
 -3.60821082e+01  8.68781069e+01  1.67819918e+01  6.76497065e+00
 -4.41334661e+01 -3.76136240e+01  3.34719873e+01  3.70726950e+00
  1.45637608e+01 -1.63779877e+01  4.15495396e+01  3.04251594e+01
  4.25791187e+01  4.28367530e+01  8.98702116e+00 -4.52435684e+01
  2.52927563e+01  1.13416508e+01  2.26455733e+01 -2.95860812e+01
 -1.43605527e+01  4.82727906e+01 -1.48771867e+01 -2.21823114e+01
  2.92816433e+00 -9.93626525e+00]
```

```
new_m=pd.DataFrame(m)
```

```
df = pd.DataFrame( new_svd_2000,index = new_m.index)
df.head()
```

	0	1	2	3	4	
0	280007.551649	195668.737325	2061.493735	4061.792719	1328.264109	-112.62814
1	141508.312658	-63344.894443	32049.286009	49022.070974	-4082.696128	2174.98262
2	108030.642411	-42526.753420	-12656.200630	-4958.577672	-1347.233952	-5379.21113
3	98453.513249	-34349.971071	-9544.318713	3848.909959	-1079.547485	-3894.34274
4	77611.323392	-22612.605758	489.155313	-5772.831384	1848.507773	1509.63193

5 rows × 150 columns

```
df.insert (0, "features",top_features)
```

```
df
```

	features	0	1	2	3	
0	students	280007.551649	195668.737325	2061.493735	4061.792719	1328.2
1	school	141508.312658	-63344.894443	32049.286009	49022.070974	-4082.6
2	learning	108030.642411	-42526.753420	-12656.200630	-4958.577672	-1347.2
3	classroom	98453.513249	-34349.971071	-9544.318713	3848.909959	-1079.5
4	not	77611.323392	-22612.605758	489.155313	-5772.831384	1848.5
5	learn	86358.819750	-31180.226150	-1245.298084	-6855.424063	-10668.5
6	help	74146.529359	-26953.434537	-14296.910372	-1264.092976	3506.6
7	many	74533.207286	-30130.215017	13967.044058	-6733.101726	1746.6
8	need	60591.060859	-24165.682712	-7763.381474	-1807.007671	1793.9
9	work	55844.096638	-20788.968099	-3910.213498	-3195.345050	-1320.6
10	come	58630.972809	-21108.988361	19240.616390	-12923.868251	-7625.1
11	use	50319.468549	-18436.424649	-11424.612181	-486.877752	4816.1
12	able	48573.901744	-19478.213671	-9824.455329	-202.733619	3641.8
13	love	54767.789525	-15547.050058	-5757.291213	-3285.951459	-2865.9
14	day	48980.621525	-8443.780256	4632.983025	-8925.512010	-10359.0
15	also	36936.302411	-10183.984009	-5836.931313	-585.371492	4604.4
16	class	41212.004937	-13673.104103	-1421.008444	590.219558	-762.0
17	make	35097.977953	-7629.270777	-5137.103742	-959.029663	-1481.8
18	new	37670.960685	-6493.905527	-2837.650837	-2240.561296	-3569.5
19	year	37225.924955	-7548.078373	4444.747101	-11012.855463	-3488.8
20	one	31871.386599	-7148.262200	2585.506069	-3948.762721	-1606.6
21	would	40803.428795	-11628.594508	-9355.527514	27.552891	2792.0
22	time	32114.164074	-8479.490933	-4312.931431	-1016.031066	1080.2
23	student	25492.462727	1996.354544	-2186.411506	-826.362668	602.5
24	want	40606.810167	-12490.046359	-3061.506906	-1741.998865	-3783.3
25	skills	36394.337196	-10094.958381	-9806.408416	520.531377	5352.3
26	grade	35085.658043	-15248.582549	2902.936045	-3242.905141	-924.5
27	reading	52695.633044	-18269.017881	-12048.907710	539.457859	7683.9
28	get	30137.587407	-8078.411280	-3022.473740	-1050.684043	-249.5
29	every	30355.177808	-4449.553160	2705.889322	-104.145047	-8059.9
...	...	...	...	...	...	
1970	born	495.662870	-263.364828	61.451245	9.570554	-28.2
1971	latest	553.627711	164.426303	110.999901	15.746945	50.5

1971	latest	553.627711	-164.426303	-110.990901	-15.746945	50.5
1972	thrilled	562.265703	-182.496509	-40.073119	-31.459715	-38.2
1973	frustrated	516.289870	-172.705982	-79.503779	16.668202	32.3
1974	erasers	420.438435	-102.579066	-114.220278	3.282658	59.0
1975	lay	460.296677	-150.176464	-89.159515	-6.923839	-10.8
1976	chances	495.652211	-116.227443	-57.909947	-17.053856	-6.4
1977	limits	492.245933	-153.377291	-3.466743	-18.454254	3.0
1978	printed	444.936095	-142.858060	-108.082127	10.628727	65.1
1979	mexico	445.587446	-221.476570	182.475142	-21.689998	-29.3
1980	discovering	502.252521	-135.095238	-93.923768	14.489666	-65.4
1981	nannanmath	275.233977	-6.455609	-115.462439	21.637269	38.4
1982	asian	377.112562	-244.751589	157.554356	-53.555228	25.2
1983	amazed	504.391236	-168.102618	4.818020	0.390064	-103.1
1984	houses	546.561825	-165.879694	232.150360	-242.247722	-30.0
1985	designs	444.872664	-203.874320	-106.054171	0.032575	37.0
1986	14	503.432187	-271.689339	104.327997	-48.854972	-14.2
1987	fullest	547.686630	-169.814419	-84.753628	-12.805573	-25.0
1988	multitude	541.873447	-236.348654	7.250693	-9.356082	27.1
1989	rarely	535.709020	-183.593422	70.502488	-93.876838	-11.3
1990	figure	430.368749	-116.644170	-81.677589	18.926455	-28.1
1991	product	434.995165	-150.336957	-81.614187	-2.081733	23.3
1992	finished	433.036368	-128.820702	-63.940680	-2.413995	38.0
1993	hunger	537.396841	-125.828662	60.467201	-22.531808	-55.0
1994	item	457.194073	-116.337303	-88.651270	-7.698261	16.6
1995	inclusive	625.643234	-158.829231	46.823705	-122.489418	-51.5
1996	evidence	391.783290	-124.482307	-120.198948	1.515019	66.3
1997	deserves	395.101762	9.340469	-34.000950	4.218788	-128.2
1998	subtraction	360.371689	-80.115075	-159.490243	20.398075	103.7
1999	builds	357.582421	-63.551533	-89.933011	1.964319	34.4

2000 rows × 151 columns

```
ind = list(top_features)
print(len(ind))
print(ind[:10])
```

```
2000
['students', 'school', 'learning', 'classroom', 'not', 'learn', 'help', 'many', 'need
```

```
vector_dict = dict()
cnt = 0

for i in new_svd_2000:

    vector_dict[ ind[cnt] ] = i
    cnt += 1
```

```
def show( text_data ):
    sentence = []

    for sen in tqdm( text_data.values ):
        fsentence = []

        for w in sen.split():
            for cw in w.split():

                if cw.isalpha():
                    fsentence.append( cw.lower() )
                else:
                    continue
        sentence.append( fsentence )
    return sentence
```

```
import numpy as np
```

```
def avgw2v( data, words ):
    sentV = []

    for sent in tqdm( data ):

        svec = np.zeros(150)
        cnw = 0

        for w in sent:
            if w in words:
                vec = vector_dict[ w ]
                svec += vec
                cnw += 1

        if cnw != 0:
            svec /= cnw
        sentV.append( svec )

    return sentV
```

```
top_150=top_features[:150]
```

```
train_final = show( S_train['text'] )

print( len( train_final ) )
```

```
100%|██████████| 46900/46900 [00:03<00:00, 13623.35it/s]46900
```

```
test_final = show( S_test['text'] )

print( len( test_final ) )
```

```
100%|██████████| 33000/33000 [00:02<00:00, 12110.78it/s]33000
```

```
cv_final = show( S_cv['text'] )

print( len( train_final ) )
```

```
100%|██████████| 20100/20100 [00:01<00:00, 13375.69it/s]46900
```

```
w2v_vectors_train= np.asarray( avgw2v( train_final, top_150 ) )

w2v_vectors_train.shape
```

```
100%|██████████| 46900/46900 [00:13<00:00, 3515.33it/s]
(46900, 150)
```

```
w2v_vectors_test= np.asarray( avgw2v( test_final, top_150 ) )

w2v_vectors_test.shape
```

```
100%|██████████| 33000/33000 [00:09<00:00, 3539.79it/s]
(33000, 150)
```

```
w2v_vectors_cv= np.asarray( avgw2v( cv_final, top_150 ) )

w2v_vectors_cv.shape
```

```
100%|██████████| 20100/20100 [00:05<00:00, 3569.64it/s]
(20100, 150)
```

```
print(w2v_vectors_train.shape)
print(w2v_vectors_test.shape)
print(w2v_vectors_cv.shape)
```

```
(46900, 150)
(33000, 150)
(20100, 150)
```

VECTORIZING TRAIN,TEST and CV data for essay

2.4 Merge the features from step 3 and step 4

```
from scipy.sparse import hstack
S_best_feat_train= hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one
print(S_best_feat_train.shape)
```

```
(46900, 258)
```

```
S_best_feat_test= hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hc
print(S_best_feat_test.shape)
```

```
(33000, 258)
```

```
S_best_feat_cv= hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_one_hot_cv,t
print(S_best_feat_cv.shape)
```

```
(20100, 258)
```

2.5 Apply XGBoost on the Final Features from the above section

```
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, auc, roc_auc_score, roc_curve
from xgboost import XGBClassifier
```

GRID SEARCH TO FIND BEST HYPERPARAMETERS

```
estimators = [10,50,100,150,200,300,500,1000]
Depths = [2,3,4,5,6,7,8,9,10]
param_grid = {'n_estimators': estimators, 'max_depth':Depths }
XGB = XGBClassifier(booster='gbtree')
xgb1 = GridSearchCV(XGB, param_grid,scoring = 'roc_auc',cv=3,n_jobs = -1,pre_dispatch=2)
xgb1.fit(S_best_feat_train, y_train)
```

```
GridSearchCV(cv=3, error_score='raise-deprecating',
                estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                        colsample_bylevel=1, colsample_bynode=1,
                                        colsample_bytree=1, gamma=0,
                                        learning_rate=0.1, max_delta_step=0,
                                        max_depth=3, min_child_weight=1,
                                        missing=None, n_estimators=100, n_jobs=1,
                                        nthread=None, objective='binary:logistic',
                                        random_state=0, reg_alpha=0, reg_lambda=1,
                                        scale_pos_weight=1, seed=None, silent=None,
                                        subsample=1, verbosity=1),
                iid='warn', n_jobs=-1,
                param_grid={'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10],
                            'n_estimators': [10, 50, 100, 150, 200, 300, 500,
                                             1000]},
                pre_dispatch=2, refit=True, return_train_score=False,
                scoring='roc_auc', verbose=0)
```

best hyperparameters are best_depth=3 and n_estimators=100

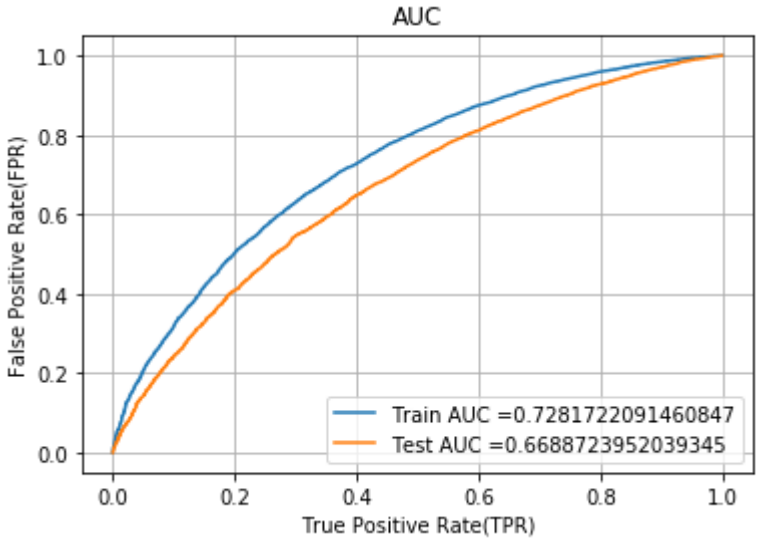
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the pos
    # not the predicted outputs

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

```
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

```
from sklearn.metrics import roc_curve, auc
clf_1 = XGBClassifier(booster='gbtree',max_depth=3,n_estimators=100)
clf_1.fit(S_best_feat_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positiv
# not the predicted outputs
y_train_pred = batch_predict(clf_1,S_best_feat_train)
y_test_pred = batch_predict(clf_1, S_best_feat_test)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```

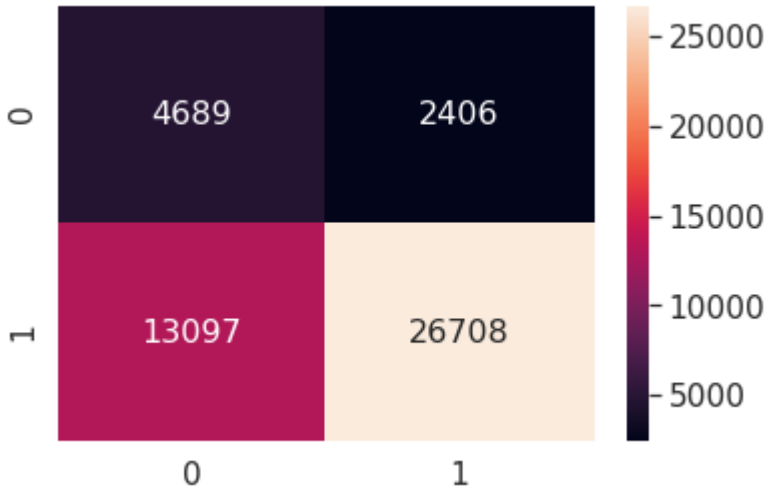



```
def prediction(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(fpr*(1-tpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Confusion matrix for train data

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, prediction(y_train_pred, tr_threshold
train_fpr, train_tpr)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

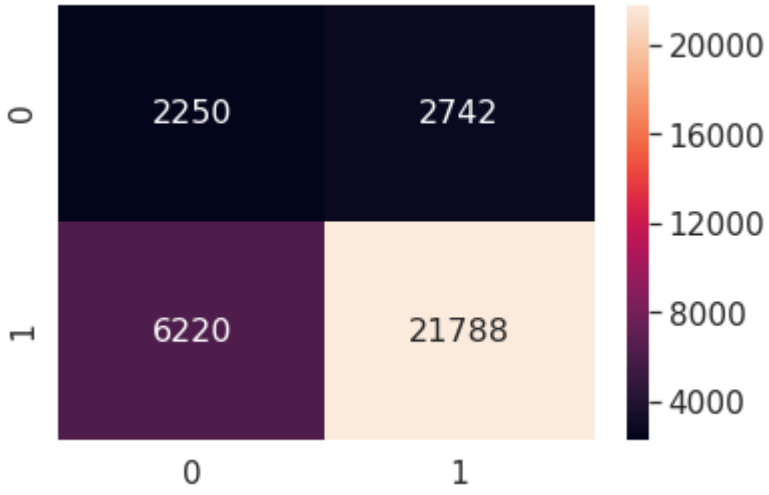
```
the maximum value of tpr*(1-fpr) 0.4451817869336412 for threshold 0.843
<matplotlib.axes._subplots.AxesSubplot at 0x7fb1960bbcf8>
```



Confusion matrix for test data

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test, prediction(y_test_pred, tr_thresholds,
train_fpr, train_tpr)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4451817869336412 for threshold 0.843
<matplotlib.axes._subplots.AxesSubplot at 0x7fb194834c18>
```



3. Conclusion

```
1.From Test confusion matrix we can observe that model is neither overfit or underfit.
2.Even with only 150 dimension the model classified fairly well.
3.Unlike PCA TruncatedSVD worked on sparse matrix.
4.The train AUC value is 0.7281 and test AUC is 0.6688.
5.Time complexity was less during training model as compared to models in 9th assignment.
```