Document Created Date:22-07-2019

Image Source: https://www.google.com/search?biw=1366&bih=637&tbm=isch&sxsrf=ACYBGNTtmKSa8sZ9msdRhqKngmdQh_XDkg%3A1577293927022&sa=1
wiz-img.......35i39j0i24j0i10i24.TlwH9NXuj1s#imgrc=ffQstrgGmatx7M (https://www.google.com/search?biw=1366&bih=637&tbm=isch&sxsrf=ACYBGNTtmKSa8sZ9msdRhqKngmdQh_XDkg%3A1577293927022&sa=1
wiz-img.......35i39j0i24j0i10i24.TlwH9NXuj1s#imgrc=ffQstrgGmatx7M):

Grasping the data and datasource - DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	
`project_id`	A unique identifier for the proposed project. **Exampl**
`project_title`	Title of the projec • `Art Will Make Y` • `First`
`project_grade_category`	Grade level of students for which the project is targeted. One of t enumera • `Grad` • `G` • `G` • `Gr`
`project_subject_categories`	One or more (comma-separated) subject categories for the pro following enumerated li • `Applied` • `Care` • `Health` • `History` • `Literacy &` • `Math` • `Music &` • `Spec` • • `Music &` • `Literacy & Language, Math`
`school_state`	State where school is located ([Two-letter U.S. (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos) **E**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories fo • • `Literature & Writing, Social`
`project_resource_summary`	An explanation of the resources needed for the projec • `My students need hands on literacy materials t sensory nee`
`project_essay_1`	First applic
`project_essay_2`	Second applic
`project_essay_3`	Third applic
`project_essay_4`	Fourth applic
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2( 12:`
`teacher_id`	A unique identifier for the teacher of the proposed proje `bdf8baa8fedef6bfeec7ae4`

Feature	
`teacher_prefix`	Teacher's title. One of the following enumer• • • • • • •
`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. Each line in this file represents a resource required by a project:

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

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

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

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

Notes on the Essay Data

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

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_3:** "Close by sharing why your project will make a difference"

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

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

```
Step by Step Procedure
```

- Understanding the Businessreal world problem
- Loading the data
- Preprocessing the data(based on the type of data = categorical , text, Numarical)
- Preprocessing data includes (removing outliers, impute missung values, cleaning data,etc..)
- Split the data into train, cv, test
- Vectorization data (one hot encoding)
- Vectorizing text data
- Normalizing
- Contactinating all the type of features(cat + text)
- Caluculating the idf values,
- sorting vocab based on idf value and selecting top 2000 words,
- Applying TruncatedSVD on co-occurance matrix for dimention reduction
- Ploting n_components v/s Explained Variance
- Sentiment scores for each essay
- Merge matrices
- Apply XGBoost on the Final Features
- Hyperparameter tuning to find th best estimator(GridSearchCV)
- Train the XGBoost model using best hyperparameter and ploting auc roc-curve
- Plot Confusion Matrix
- Observation on overall model performences
- Ploting the performences by tableu format.

```
C:\Users\Ramesh Battu> import required libraries
```

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
```

1.1 Reading Data

In [9]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [10]:

```
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 [11]:

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

#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)

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

In [12]:

```
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[12]:

	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

1.1.1 preprocessing of `project_subject_categories`

In [13]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
catogories = list(project_data['project_subject_categories'].values)

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
        if 'The' in j.split(): # this will split each of the catogory based on space "Ma
            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:"Ma
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        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]))
```

1.1.2 preprocessing of `project_subject_subcategories`

In [14]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pytho

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
        if 'The' in j.split(): # this will split each of the catogory based on space "Ma
            j=j.replace('The','') # if we have the words "The" we are going to replace i
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Ma
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

1.1.3 prepocessing of `school_state`

In [15]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pytho
school_state_catogories = list(project_data['school_state'].values)
cat_list = []
for i in school_state_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
        if 'The' in j.split(): # this will split each of the catogory based on space "Ma
            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:"Ma
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
project_data['school_state'] = cat_list

from collections import Counter
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())

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

1.1.4 Preprocessing of `teacher_prefix`

In [152]:

```
# citation code :https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['teacher_prefix'].value_counts().index[
teacher_prefix_catogories = list(project_data['teacher_prefix'].values)
# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-sciki
# To convert the data type object to unicode string : used """astype('U')""" code from
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in teacher_prefix_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
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            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:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
        temp =temp.replace('.', '') # we are removing dot(.)
    cat_list.append(temp.strip())

project_data['teacher_prefix'] = cat_list

from collections import Counter
my_counter = Counter()
for word in project_data['teacher_prefix'].values:
    word = str(word)
    my_counter.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv: kv
```

In [153]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_tp = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), low

teacher_prefix_one_hot = vectorizer_tp.fit_transform(project_data['teacher_prefix'].val
print("vectorizer of project_subject_subcategories feature names :", vectorizer_tp.get_
```

```
vectorizer of project_subject_subcategories feature names : ['Dr', 'Teache
r', 'Mr', 'Ms', 'Mrs']
```

1.1.5 Preprocessing of `project_grade_category`

In [154]:

```
project_grade_catogories = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in project_grade_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
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            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:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
        temp = temp.replace('-','_') # we are replaceing '-' with '_'
    cat_list.append(temp.strip())

project_data['project_grade_category'] = cat_list

#link : https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['project_grade_category'].value_counts(

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grade_category_dict = dict(my_counter)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), k
```

1.2. Text Preprocessing

1.2.1 Text Preprocessing of essay

In [18]:

```
# 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 [0]:

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

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	projec
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

In [19]:

```python
# printing some random reviews
print(project_data['essay'].values[20])
print("="*50)
print(project_data['essay'].values[120])
print("="*50)
print(project_data['essay'].values[1020])
print("="*50)
print(project_data['essay'].values[20020])
print("="*50)
print(project_data['essay'].values[99920])
print("="*50)
```

```
Throughout this school year, I hope to enable my students to develop a l
ove for reading, discovering, and creating. Students in my classroom hav
e the opportunity to explore learning through large group activities, sm
all group learning times, and interactive play with other students and t
eachers.My school is a Pre-K through 8 school with teachers and staff wh
o are determined to help children in Detroit succeed. Students in my cla
ssroom are just beginning their education in Pre-K and and I strive to e
nable my students to shine brightly as they learn using the High Scope C
urriculum.The shopping carts would be a wonderful addition to our dramat
ic play area. With them, the children will be able to work on their soci
al skills. They will learn about sharing, purchasing items, and the cost
of different items. Similarly, the parachute will be used to encourage c
ooperative play. The children will have fun learning and exercising. I'm
looking forward to adding the beads to our art area in the classroom to
encourage both creativity as well as fine motor skills as the children w
ork on stringing them.Donations to this project will allow me to encoura
ge my students' curiosity and develop a love for learning in them.\r\nWi
th these resources, my students will start off on the right path in thei
r education
==================================================
Every afternoon we extend our learning through creative play. Creative p
repares kids to be next generation innovators!My students' location does
not define their ability or determine their future.  Our city has been v
oted the \"Most Dangerous\" and \"Poorest\" city in the United States co
nsistently in the past 30 years.  Fortunately these labels are not actua
l descriptions of who we are.  My students are innocent, hardworking, en
ergetic, inquisitive and zealous for knowledge.  They are young leaders
fighting for a 21st century education. \r\nIn my school, 100% of our eag
er students receive free breakfast and lunch.  100% of our dedicated stu
dents are thirsty to learn.  100% of our teachers are driving our studen
ts to grow beyond a year's growth.  We are in the fight and we will win.
We are in need of resources that will spark our creativity and open our
minds to possibilities. The light table will supply a different way to v
iew material. While the kinetic sand and water bead will allow them to u
se their senses to create new artifacts that extend their thinking and i
magine boundless ideas.   These tools will encourage discussion and team
building techniques.These resources will fuel our learning as they open
their minds to what's possible. They will encourage my students to take
change, problem solve, and collaborate. They will be come creative think
ers and doers.  It will also help my students cope in their everyday liv
es.
==================================================
Over 25% of Kindergarten have significant poor attention and focus.  Sen
sory tools can help kids focus, so they can learn.  Movement and physica
l activity are imperative to brain development, academic performance and
self-regulation.These tools increase teaching time and decrease redirect
ion time.Our students are from a school district where about 50% of the
```

kids qualify for the free lunch program.   Many of the kids that need se
nsory tools are boys. Our kids need to move to pay attention. Providin
g the kids with sensory tools can give them a well deserved advantage in
the classroom. Many of our kids are in under served population groups,
whose families struggle to put food on the table and have few resources
to support their child's education The kids need our help to increase f
ocus and attention in the classroom. The teachers need easily accessibl
e tools they can use during instruction to provide needed movement to ke
ep kids focused and learning.The sensory toolbox items will be available
to the students as needed.   When the teacher sees that \"Susie\" is off
task, she will walk around to her and offer her a fidget toy from the fi
dget basket.   Susie chooses the squishy stress ball and starts squeezin
g it. Soon, Susie is back on task and answering the next question corre
ctly.   When the teacher sees Joey tapping his foot, slouching in his ch
air or standing frequently at his desk, she carries a move and sit cushi
on to Joey to sit on.   Joey proceeds to wiggle on the seat cushion and
is able to stay in his seat, yet receive the sought after movement. Bot
h these examples demonstrate how attention and focus are improved withou
t the use of disciplinary actions or yelling. Using this proactive appr
oach increases confidence, intelligence, and over all learning.Donations
towards my project to fund sensory tools for the classrooms will empower
teachers to make a difference in how well students can attend and focus,
while at the same time reducing the need for disciplinary actions relate
d to poor attention and focus in the classroom. Its a win-win situatio
n. Teachers can teach, and children can learn by keeping them moving an
d fidgeting.
===
I am a second year teacher and I have the pleasure of teaching a kinderg
arten class that is filled with 21 sweet South Carolina students. Part
of my job as a kindergarten teacher is to make sure my students have a s
afe and fun environment in which to learn so they will have a good outlo
ok about school. I have the privilege to help change the lives of childr
en and I want nothing but the best for my class. My students are between
5 and 6 years old and they are learning some of the hardest things they
will ever learn: reading, writing, and sharing. They love being at schoo
l and having a fun environment to learn in. My students come from all wa
lks of life and I want each student to have the best chance to succeed.I
am a kindergarten teacher at a public charter school that will be open f
or its third year this month. My students last year inspired me to crea
te this project, since everyday I saw some sit and stand instead of play
ing and having fun. My students love to play football and last year the
y played it using their imagination, since we did not have a football.
There is a big need for playground equipment that allows them to move an
d play. We have a basketball goal but the balls are needing replaced.
The bowling ball set and hula hoops will be great for those that don't w
ant to play a contact game. The mobile cart will be a lifesaver with mo
ving the equipment to and from the playground each day. \r\nMost of all
my students will not stand around during recess and not play when they g
et these wonderful items!Mrs.Mrs.
===
I have a large population of Haitian students in our beautiful classroom
in P.H. Elementary in Florida. Many enter my class struggling with the E
nglish language and we work very hard to keep up with the Common Core St
andards. I work in a Title I school and my students live in high povert
y. It's a very old school building and needles to say supplies of any k
ind are short in demand. Luckily love and determination are in abundance
in my young hard workers.We don't have such beautiful supplies that can
quickly engage the mind of the learner and show them that there are so m
any ways to learn through play. We have Center-Time in our classrooms wh
ich must show them to learn independently and these types of supplies wo
uld push them to create exemplary center work. These games will allow th

```
em to grow academically while actually playing a game. What is now unden
iably clear in the 21st century is that play is essential, vital, critic
al, and fundamental to a child’s social, emotional, physical, and intell
ectual development. These types of supplies hold these words to be true
in a learning classroom.Mrs.Mrs.
==================================================
```

In [20]:

```
# 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 [22]:

```
sent = decontracted(project_data['essay'].values[20020])
print(sent)
print("="*80)
```

```
I am a second year teacher and I have the pleasure of teaching a kindergar
ten class that is filled with 21 sweet South Carolina students.  Part of m
y job as a kindergarten teacher is to make sure my students have a safe an
d fun environment in which to learn so they will have a good outlook about
school. I have the privilege to help change the lives of children and I wa
nt nothing but the best for my class. My students are between 5 and 6 year
s old and they are learning some of the hardest things they will ever lear
n: reading, writing, and sharing. They love being at school and having a f
un environment to learn in. My students come from all walks of life and I
want each student to have the best chance to succeed.I am a kindergarten t
eacher at a public charter school that will be open for its third year thi
s month.  My students last year inspired me to create this project, since
everyday I saw some sit and stand instead of playing and having fun.  My s
tudents love to play football and last year they played it using their ima
gination, since we did not have a football.  There is a big need for playg
round equipment that allows them to move and play.  We have a basketball g
oal but the balls are needing replaced.  The bowling ball set and hula hoo
ps will be great for those that do not want to play a contact game.  The m
obile cart will be a lifesaver with moving the equipment to and from the p
layground each day.  \r\nMost of all my students will not stand around dur
ing recess and not play when they get these wonderful items!Mrs.Mrs.
==========================================================================
======
```

In [23]:

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

```
I am a second year teacher and I have the pleasure of teaching a kindergar
ten class that is filled with 21 sweet South Carolina students.  Part of m
y job as a kindergarten teacher is to make sure my students have a safe an
d fun environment in which to learn so they will have a good outlook about
school. I have the privilege to help change the lives of children and I wa
nt nothing but the best for my class. My students are between 5 and 6 year
s old and they are learning some of the hardest things they will ever lear
n: reading, writing, and sharing. They love being at school and having a f
un environment to learn in. My students come from all walks of life and I
want each student to have the best chance to succeed.I am a kindergarten t
eacher at a public charter school that will be open for its third year thi
s month.  My students last year inspired me to create this project, since
everyday I saw some sit and stand instead of playing and having fun.  My s
tudents love to play football and last year they played it using their ima
gination, since we did not have a football.  There is a big need for playg
round equipment that allows them to move and play.  We have a basketball g
oal but the balls are needing replaced.  The bowling ball set and hula hoo
ps will be great for those that do not want to play a contact game.  The m
obile cart will be a lifesaver with moving the equipment to and from the p
layground each day.    Most of all my students will not stand around durin
g recess and not play when they get these wonderful items!Mrs.Mrs.
```

In [24]:

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

```
I am a second year teacher and I have the pleasure of teaching a kindergar
ten class that is filled with 21 sweet South Carolina students Part of my
job as a kindergarten teacher is to make sure my students have a safe and
fun environment in which to learn so they will have a good outlook about s
chool I have the privilege to help change the lives of children and I want
nothing but the best for my class My students are between 5 and 6 years ol
d and they are learning some of the hardest things they will ever learn re
ading writing and sharing They love being at school and having a fun envir
onment to learn in My students come from all walks of life and I want each
student to have the best chance to succeed I am a kindergarten teacher at
a public charter school that will be open for its third year this month My
students last year inspired me to create this project since everyday I saw
some sit and stand instead of playing and having fun My students love to p
lay football and last year they played it using their imagination since we
did not have a football There is a big need for playground equipment that
allows them to move and play We have a basketball goal but the balls are n
eeding replaced The bowling ball set and hula hoops will be great for thos
e that do not want to play a contact game The mobile cart will be a lifesa
ver with moving the equipment to and from the playground each day Most of
all my students will not stand around during recess and not play when they
get these wonderful items Mrs Mrs
```

In [25]:

```python
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
            "as","at","be","because","been","before","being","below","between","both",
            "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
            "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
            "here","hers","herself","him","himself","his","how","i","if","in","into","
            "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
            "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
            "out","over","own","re","s","same","shan","shan't","she","she's","should",
            "so","some","such","t","than","that","that'll","the","their","theirs","the
            "these","they","this","those","through","to","too","under","until","up","v
            "we","were","weren","weren't","what","when","where","which","while","who",
            "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
            "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
            "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
            "what's","when's","where's","who's","why's","would","able","abst","accorda
            "across","act","actually","added","adj","affected","affecting","affects","
            "along","already","also","although","always","among","amongst","announce",
            "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
            "around","aside","ask","asking","auth","available","away","awfully","b","b
            "becoming","beforehand","begin","beginning","beginnings","begins","behind"
            "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
            "co","com","come","comes","contain","containing","contains","couldnt","dat
            "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsew
            "especially","et","etc","even","ever","every","everybody","everyone","ever
            "f","far","ff","fifth","first","five","fix","followed","following","follow
            "found","four","furthermore","g","gave","get","gets","getting","give","giv
            "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
            "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
            "importance","important","inc","indeed","index","information","instead","i
            "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
            "later","latter","latterly","least","less","lest","let","lets","like","lik
            "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
            "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
            "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
            "neither","never","nevertheless","new","next","nine","ninety","nobody","no
            "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
            "omitted","one","ones","onto","ord","others","otherwise","outside","overal
            "particular","particularly","past","per","perhaps","placed","please","plus
            "potentially","pp","predominantly","present","previously","primarily","pro
            "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
            "recently","ref","refs","regarding","regardless","regards","related","rela
            "resulted","resulting","results","right","run","said","saw","say","saying"
            "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
            "shes","show","showed","shown","showns","shows","significant","significant
            "six","slightly","somebody","somehow","someone","somethan","something","so
            "somewhere","soon","sorry","specifically","specified","specify","specifyin
            "sub","substantially","successfully","sufficiently","suggest","sup","sure"
            "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
            "therein","there'll","thereof","therere","theres","thereto","thereupon","t
            "thou","though","thoughh","thousand","throug","throughout","thru","thus","
            "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
            "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
            "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
            "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
            "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
            "who'll","whomever","whos","whose","widely","willing","wish","within","wit
            "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
```

```
             "appreciate","appropriate","associated","best","better","c'mon","c's","can
             "consequently","consider","considering","corresponding","course","currentl
             "entirely","exactly","example","going","greetings","hello","help","hopeful
             "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","p
             "secondly","sensible","serious","seriously","sure","t's","third","thorough
             "wonder"]
```

In [26]:

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

```
Wall time: 0 ns

100%|█████████████████████████████████| 109248/109248 [05:55<00:00, 307.31
it/s]
```

In [27]:

```
# after preprocesing
preprocessed_essays[20020]
```

Out[27]:

```
'year teacher pleasure teaching kindergarten class filled 21 sweet south c
arolina students job kindergarten teacher students safe fun environment le
arn good outlook school privilege change lives children class students 5 6
years learning hardest things learn reading writing sharing love school fu
n environment learn students walks life student chance succeed kindergarte
n teacher public charter school open year month students year inspired cre
ate project everyday sit stand playing fun students love play football yea
r played imagination football big playground equipment move play basketbal
l goal balls needing replaced bowling ball set hula hoops great play conta
ct game mobile cart lifesaver moving equipment playground day students sta
nd recess play wonderful items'
```

1.2.2 Text Preprocessing of `project_title`

In [28]:

```
print(project_data['project_title'].tail(1))
```

```
78306    News for Kids
Name: project_title, dtype: object
```

In [29]:

```
# printing some random title texts
print(project_data['project_title'].values[20])
print('--'*19)
print(project_data['project_title'].values[120])
print('--'*19)
print(project_data['project_title'].values[1920])
print('--'*19)
print(project_data['project_title'].values[99920])
print('--'*19)
```

```
Pre-K Classroom Materials
--------------------------------------
Empowering Kindergarteners
--------------------------------------
A Classroom Fit for all Learners
--------------------------------------
Turn That TV Game Off and Turn on the Learning!
--------------------------------------
```

In [30]:

```
# 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 [31]:

```
sent = decontracted(project_data['project_title'].values[99920])
print(sent)
print("="*120)
```

```
Turn That TV Game Off and Turn on the Learning!
==========================================================================
==============================================
```

In [32]:

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

```
Turn That TV Game Off and Turn on the Learning
```

In [33]:

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

```
Turn That TV Game Off and Turn on the Learning
```

In [34]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
             "as","at","be","because","been","before","being","below","between","both",
             "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
             "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
             "here","hers","herself","him","himself","his","how","i","if","in","into","
             "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
             "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
             "out","over","own","re","s","same","shan","shan't","she","she's","should",
             "so","some","such","t","than","that","that'll","the","their","theirs","the
             "these","they","this","those","through","to","too","under","until","up","v
             "we","were","weren","weren't","what","when","where","which","while","who",
             "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
             "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
             "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
             "what's","when's","where's","who's","why's","would","able","abst","accorda
             "across","act","actually","added","adj","affected","affecting","affects","
             "along","already","also","although","always","among","amongst","announce",
             "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
             "around","aside","ask","asking","auth","available","away","awfully","b","b
             "becoming","beforehand","begin","beginning","beginnings","begins","behind"
             "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
             "co","com","come","comes","contain","containing","contains","couldnt","dat
             "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsew
             "especially","et","etc","even","ever","every","everybody","everyone","ever
             "f","far","ff","fifth","first","five","fix","followed","following","follow
             "found","four","furthermore","g","gave","get","gets","getting","give","giv
             "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
             "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
             "importance","important","inc","indeed","index","information","instead","i
             "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
             "later","latter","latterly","least","less","lest","let","lets","like","lik
             "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
             "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
             "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
             "neither","never","nevertheless","new","next","nine","ninety","nobody","no
             "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
             "omitted","one","ones","onto","ord","others","otherwise","outside","overal
             "particular","particularly","past","per","perhaps","placed","please","plus
             "potentially","pp","predominantly","present","previously","primarily","pro
             "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
             "recently","ref","refs","regarding","regardless","regards","related","rela
             "resulted","resulting","results","right","run","said","saw","say","saying"
             "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
             "shes","show","showed","shown","showns","shows","significant","significant
             "six","slightly","somebody","somehow","someone","somethan","something","so
             "somewhere","soon","sorry","specifically","specified","specify","specifyin
             "sub","substantially","successfully","sufficiently","suggest","sup","sure"
             "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
             "therein","there'll","thereof","therere","theres","thereto","thereupon","t
             "thou","though","thoughh","thousand","throug","throughout","thru","thus","
             "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
             "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
             "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
             "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
             "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
             "who'll","whomever","whos","whose","widely","willing","wish","within","wit
             "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
```

```
               "appreciate","appropriate","associated","best","better","c'mon","c's","can
               "consequently","consider","considering","corresponding","course","currentl
               "entirely","exactly","example","going","greetings","hello","help","hopeful
               "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","p
               "secondly","sensible","serious","seriously","sure","t's","third","thorough
               "wonder"]
```

In [35]:

```
%%time
# Combining all the above stundents
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 = sent.replace('!', ' ')
    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 [00:12<00:00, 8822.72
it/s]

Wall time: 12.4 s
```

In [36]:

```
preprocessed_project_title[99999]
```

Out[36]:

```
'turning flexible seating sixth grade class journey freedom'
```

1.3. Numerical normalization

1.3.1 normalization_price

In [38]:

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

Out[38]:

```
(109248, 20)
```

In [39]:

```
# later merged dataframes
project_data.head(1)
```

Out[39]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Dat
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016 04-2 00:27:3

In [40]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(1,-1)
price_normalize = Normalizer()
price_normalizer = price_normalize.fit_transform(project_data['price'].values.reshape(1
price_normalizer = price_normalizer.T
print(price_normalizer)
print("-------------------------------------------------------")
print("shape of price_normalizer:", price_normalizer.shape)
```

```
[[4.63560392e-03]
 [1.36200635e-03]
 [2.10346002e-03]
 ...
 [2.55100471e-03]
 [1.83960046e-03]
 [3.51642253e-05]]
-------------------------------------------------------
shape of price_normalizer: (109248, 1)
```

1.3.2 teacher number of previously posted projects normalization

In [41]:

```
project_data['teacher_number_of_previously_posted_projects'].values
```

Out[41]:

```
array([53,  4, 10, ...,  0,  1,  2], dtype=int64)
```

In [42]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
teacher_number_of_previously_posted_projects_normalize = Normalizer()
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
print(teacher_number_of_previously_posted_projects_normalizer)
print("="*25)
print("Shape of teacher_number_of_previously_posted_projects_normalizer :", teacher_num
```

```
[[0.00535705]
 [0.00040431]
 [0.00101076]
 ...
 [0.        ]
 [0.00010108]
 [0.00020215]]
=========================
Shape of teacher_number_of_previously_posted_projects_normalizer : (10924
8, 1)
```

1.3.3. quantity Normalization

In [44]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
normalizer = Normalizer()
quantity_normalizer = normalizer.fit_transform(project_data['quantity'].values.reshape(
print(quantity_normalizer)
print("Shape of quantity nomalizer :", quantity_normalizer.shape)
```

```
[[3.87894657e-04]
 [7.75789314e-04]
 [9.69736642e-05]
 ...
 [4.84868321e-04]
 [3.87894657e-04]
 [2.42434160e-03]]
Shape of quantity nomalizer : (109248, 1)
```

In [45]:

```
project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_project_title'] = preprocessed_project_title
```

1.3.4 spilt the data into train ,CV and test

In [46]:

```
label = project_data['project_is_approved']
project_data.drop(['project_is_approved'], axis=1, inplace=True)
# spliting the data into train , test CV
# Refrence link :https://scikit-learn.org/stable/modules/generated/sklearn.model_select
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data, label, test_size=0.33
X_train, X_cv,  y_train, y_cv    = train_test_split(X_train, y_train, test_size=0.33) #

print("Shape of X_train and y_train  :", X_train.shape, y_train.shape)
print("Shape of X_test and y_test    :", X_test.shape, y_test.shape)
print("Shape of X_cv and y_cv        :", X_cv.shape, y_cv.shape)
```

```
Shape of X_train and y_train  : (49041, 21) (49041,)
Shape of X_test and y_test    : (36052, 21) (36052,)
Shape of X_cv and y_cv        : (24155, 21) (24155,)
```

1.4. Vectorizing Categorical data

1.4.1 Vectorization of `clean_categories`

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

In [47]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=Fal

clean_categories_one_hot_train = vectorizer_cat.fit_transform(X_train['clean_categories
clean_categories_one_hot_test  = vectorizer_cat.transform(X_test['clean_categories'].va
clean_categories_one_hot_cv    = vectorizer_cat.transform(X_cv['clean_categories'].valu

print("vectorizer of project_subject_categories feature names :", vectorizer_cat.get_fe
print("----------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",clean_categories_one_hot_train.s
print("Shape of matrix after one hot encodig test  : ",clean_categories_one_hot_test.sh
print("Shape of matrix after one hot encodig cv    : ",clean_categories_one_hot_cv.shap
```

```
vectorizer of project_subject_categories feature names : ['Warmth', 'Care_
Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeed
s', 'Health_Sports', 'Math_Science', 'Literacy_Language']
----------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 9)
Shape of matrix after one hot encodig test  :  (36052, 9)
Shape of matrix after one hot encodig cv    :  (24155, 9)
```

1.4.2 Vectorization of `clean_sub_categories`

In [48]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_subcat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowerc

clean_subcategories_one_hot_train = vectorizer_subcat.fit_transform(X_train['clean_subc
clean_subcategories_one_hot_test  = vectorizer_subcat.transform(X_test['clean_subcatego
clean_subcategories_one_hot_cv    = vectorizer_subcat.transform(X_cv['clean_subcategori

print("vectorizer of project_subject_subcategories feature names :", vectorizer_subcat.
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",clean_subcategories_one_hot_trai
print("Shape of matrix after one hot encodig test  : ",clean_subcategories_one_hot_test
print("Shape of matrix after one hot encodig cv    : ",clean_subcategories_one_hot_cv.s
```

```
vectorizer of project_subject_subcategories feature names : ['Economics',
'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurric
ular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'War
mth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducati
on', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geogra
phy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Env
ironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'S
pecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 30)
Shape of matrix after one hot encodig test  :  (36052, 30)
Shape of matrix after one hot encodig cv    :  (24155, 30)
```

1.4.3 Vectorization of `school_state`

In [49]:

```
# we use count vectorizer to convert the values into one
vectorizer_ss = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lower

school_state_one_hot_train = vectorizer_ss.fit_transform(X_train['school_state'].values
school_state_one_hot_test  = vectorizer_ss.transform(X_test['school_state'].values)
school_state_one_hot_cv    = vectorizer_ss.transform(X_cv['school_state'].values)

print("vectorizer of school_state feature names :",vectorizer_ss.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",school_state_one_hot_train.shape
print("Shape of matrix after one hot encodig test  : ",school_state_one_hot_test.shape)
print("Shape of matrix after one hot encodig cv    : ",school_state_one_hot_cv.shape)
```

```
vectorizer of school_state feature names : ['VT', 'WY', 'ND', 'MT', 'RI',
'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'IA', 'I
D', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'A
L', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'P
A', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX', 'CA']
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 51)
Shape of matrix after one hot encodig test  :  (36052, 51)
Shape of matrix after one hot encodig cv    :  (24155, 51)
```

1.4.4 Vectorization of `teacher_prefix`

In [155]:

```
# we use count vectorizer to convert the values intp one hot encode
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_tp = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), low

teacher_prefix_one_hot_train = vectorizer_tp.fit_transform(X_train['teacher_prefix'].va
teacher_prefix_one_hot_test  = vectorizer_tp.transform(X_test['teacher_prefix'].values.
teacher_prefix_one_hot_cv    = vectorizer_tp.transform(X_cv['teacher_prefix'].values.as

print("Vectorizer of teacher_prefix :",vectorizer_tp.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",teacher_prefix_one_hot_train.sha
print("Shape of matrix after one hot encodig test  : ",teacher_prefix_one_hot_test.shap
print("Shape of matrix after one hot encodig cv    : ",teacher_prefix_one_hot_cv.shape)
```

```
Vectorizer of teacher_prefix : ['Dr', 'Teacher', 'Mr', 'Ms', 'Mrs']
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 5)
Shape of matrix after one hot encodig test  :  (36052, 5)
Shape of matrix after one hot encodig cv    :  (24155, 5)
```

In [52]:

```
vectorizer_cat.get_feature_names
```

Out[52]:

```
<bound method CountVectorizer.get_feature_names of CountVectorizer(analyze
r='word', binary=True, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='cont
ent',
                lowercase=False, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None,
                vocabulary=['Warmth', 'Care_Hunger', 'History_Civics',
                            'Music_Arts', 'AppliedLearning', 'SpecialNeed
s',
                            'Health_Sports', 'Math_Science',
                            'Literacy_Language'])>
```

1.4.5 Vectorization of `project_grade_categorie`

In [156]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_pgc = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.key

project_grade_category_one_hot_train = vectorizer_pgc.fit_transform(X_train['project_gr
project_grade_category_one_hot_test  = vectorizer_pgc.transform(X_test['project_grade_c
project_grade_category_one_hot_cv    = vectorizer_pgc.transform(X_cv['project_grade_cat

print("Vectorizer of project_grade_categories :",vectorizer_pgc.get_feature_names())
print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train  : ",project_grade_category_one_hot_
print("Shape of matrix after one hot encodig test   : ",project_grade_category_one_hot_
print("Shape of matrix after one hot encodig cv     : ",project_grade_category_one_hot_
```

```
Vectorizer of project_grade_categories : ['Grades9_12', 'Grades6_8', 'Grad
es3_5', 'GradesPreK_2']
---------------------------------------------------------
Shape of matrix after one hot encodig train  :  (49041, 4)
Shape of matrix after one hot encodig test   :  (36052, 4)
Shape of matrix after one hot encodig cv     :  (24155, 4)
```

1.5. Vectorizing Text and Numarical

1.5.1 Vectorization of essays_tfidf:: Train , Test and CV

In [54]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_essays_tfidf = TfidfVectorizer(preprocessed_essays, min_df=10, max_features=

essays_tfidf_train = vectorizer_essays_tfidf.fit_transform(X_train['preprocessed_essays
essays_tfidf_test  = vectorizer_essays_tfidf.transform(X_test['preprocessed_essays'].va
essays_tfidf_cv    = vectorizer_essays_tfidf.transform(X_cv['preprocessed_essays'].valu

print("Shape of matrix after one hot encodig of train : ",essays_tfidf_train.shape)
print("Shape of matrix after one hot encodig test     : ",essays_tfidf_test.shape)
print("Shape of matrix after one hot encodig cv       : ",essays_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig of train :  (49041, 5000)
Shape of matrix after one hot encodig test     :  (36052, 5000)
Shape of matrix after one hot encodig cv       :  (24155, 5000)
```

1.5.2 Vectorization of project_title tfidf ::Train, Test and Cv

In [55]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pro
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_project_title_tfidf = TfidfVectorizer(preprocessed_project_title, min_df=10,

project_title_tfidf_train = vectorizer_project_title_tfidf.fit_transform(X_train['proje
project_title_tfidf_test  = vectorizer_project_title_tfidf.transform(X_test['project_ti
project_title_tfidf_cv    = vectorizer_project_title_tfidf.transform(X_cv['project_titl

print("Shape of matrix after one hot encodig of train : ",project_title_tfidf_train.sha
print("Shape of matrix after one hot encodig test     : ",project_title_tfidf_test.shap
print("Shape of matrix after one hot encodig cv       : ",project_title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig of train :  (49041, 2111)
Shape of matrix after one hot encodig test     :  (36052, 2111)
Shape of matrix after one hot encodig cv       :  (24155, 2111)
```

1.5.3 Vectorization of `price` :: train, test and cv

In [103]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html
# Reshaping price data using array.reshape(-1, 1)
price_normalizer = Normalizer()
price_normalizer_train = price_normalizer.fit_transform(X_train['price'].values.reshape
price_normalizer_test  = price_normalizer.transform(X_test['price'].values.reshape(1,-1
price_normalizer_cv    = price_normalizer.transform(X_cv['price'].values.reshape(1,-1))

print("shape of price_normalizer_train:", price_normalizer_train.shape)
print("---------------------------")
print(price_normalizer_train)

print("shape of price_normalizer_test :", price_normalizer_test.shape)
print("---------------------------")
print(price_normalizer_test)

print("shape of price_normalizer_cv   :", price_normalizer_cv.shape)
print("---------------------------")
print(price_normalizer_cv)
```

```
shape of price_normalizer_train: (49041, 1)
---------------------------
[[0.0017128 ]
 [0.00052738]
 [0.00133911]
 ...
 [0.0006195 ]
 [0.00622391]
 [0.00297778]]
shape of price_normalizer_test : (36052, 1)
---------------------------
[[0.00173636]
 [0.00245599]
 [0.00017022]
 ...
 [0.00519457]
 [0.00457257]
 [0.00179421]]
shape of price_normalizer_cv   : (24155, 1)
---------------------------
[[0.00398983]
 [0.00302489]
 [0.00184272]
 ...
 [0.00373467]
 [0.01030215]
 [0.00143378]]
```

1.5.4 Vectorization of teacher number of previously posted projects :: train , test , cv

In [104]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html
# Reshaping price data using array.reshape(-1, 1)
tnppp_normalizer = Normalizer()
tnppp_normalizer_train = tnppp_normalizer.fit_transform(X_train['teacher_number_of_prev
tnppp_normalizer_test  = tnppp_normalizer.transform(X_test['teacher_number_of_previousl
tnppp_normalizer_cv    = tnppp_normalizer.transform(X_cv['teacher_number_of_previously_
print("shape of teacher_number_of_previously_posted_projects_normalizer_train:",tnppp_n
print("---------------------------")
print(tnppp_normalizer_train)

print("shape of teacher_number_of_previously_posted_projects_normalizer_test :",tnppp_n
print("---------------------------")
print(tnppp_normalizer_test)

print("shape of teacher_number_of_previously_posted_projects_normalizer_cv   :",tnppp_n
print("---------------------------")
print(tnppp_normalizer_cv)
```

```
shape of teacher_number_of_previously_posted_projects_normalizer_train: (4
9041, 1)
---------------------------
[[0.        ]
 [0.00343163]
 [0.00268562]
 ...
 [0.00074601]
 [0.0005968 ]
 [0.        ]]
shape of teacher_number_of_previously_posted_projects_normalizer_test : (3
6052, 1)
---------------------------
[[0.00160903]
 [0.        ]
 [0.        ]
 ...
 [0.00071512]
 [0.        ]
 [0.00035756]]
shape of teacher_number_of_previously_posted_projects_normalizer_cv   : (2
4155, 1)
---------------------------
[[0.0004296 ]
 [0.00365163]
 [0.        ]
 ...
 [0.0006444 ]
 [0.0006444 ]
 [0.        ]]
```

1.5.5 Vectorization of quantity : train, test, cv

In [111]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
normalizer = Normalizer()
quantity_normalizer_train = normalizer.fit_transform(X_train['quantity'].values.reshape
quantity_normalizer_test = normalizer.transform(X_test['quantity'].values.reshape(1,-1)
quantity_normalizer_cv   = normalizer.transform(X_cv['quantity'].values.reshape(1,-1)).

print("Shape of quantity nomalizer train :", quantity_normalizer_train.shape)
print(quantity_normalizer_train)
print("-----------------------------------------------------------------------")
print(quantity_normalizer_test)
print("Shape of quantity nomalizer test :", quantity_normalizer_test.shape)
print("-----------------------------------------------------------------------")
print(quantity_normalizer_cv)
print("Shape of quantity nomalizer cv :", quantity_normalizer_cv.shape)
```

```
Shape of quantity nomalizer train : (49041, 1)
[[0.00028972]
 [0.00028972]
 [0.00115888]
 ...
 [0.0028972 ]
 [0.00478037]
 [0.00130374]]
-----------------------------------------------------------------------
[[0.00283631]
 [0.0020021 ]
 [0.03003147]
 ...
 [0.00650682]
 [0.00083421]
 [0.00316999]]
Shape of quantity nomalizer test : (36052, 1)
-----------------------------------------------------------------------
[[0.00901331]
 [0.00083845]
 [0.00230573]
 ...
 [0.00838447]
 [0.00020961]
 [0.00167689]]
Shape of quantity nomalizer cv : (24155, 1)
```

1.6 concatinating essay + project_title

In [56]:

```
# After preprocessing concatinating essay + project_title
concat_essays_titles = (list(X_train['preprocessed_essays'])+list(X_train['preprocessed
len(concat_essays_titles)
```

Out[56]:

```
98082
```

1.6.1 Caluculating the idf values for every word in combined text

In [59]:

```
tfidf_vec = TfidfVectorizer()
tfidf_vec.fit_transform(concat_essays_titles)
```

Out[59]:

```
<98082x42148 sparse matrix of type '<class 'numpy.float64'>'
	with 4011283 stored elements in Compressed Sparse Row format>
```

In [208]:

```
idf_score = tfidf_vec.idf_
features_names = tfidf_vec.get_feature_names()
```

1.6.2 sorting vocab based on idf value in desending order

1.6.3 Selecting top 2000 words

In [198]:

```
idf_score_features = []
for i in range(len(idf_score)):
    idf_score_features.append([idf_score[i], features_names[i]])
idf_sort = idf_score_features.sort(reverse=True)
idf_score_features=idf_score_features[:2000]
```

In [ ]:

```
n = 2000
topFeat = [feat[i] for i in ind[:n]]
topIDF = [tfidfvect.idf_[i] for i in ind[:n]]
featIDF = list(zip(topFeat,topIDF))
print(featIDF[:20])
```

Preparing data for models

In [149]:

```
project_data.columns
```

Out[149]:

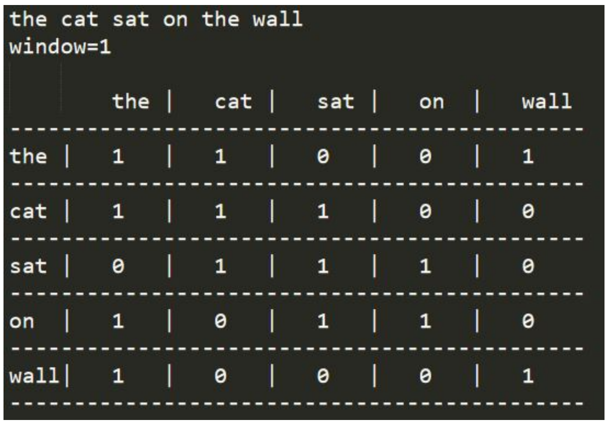
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'clean_subcategories', 'essay', 'price', 'quantity',
       'preprocessed_essays', 'preprocessed_project_title'],
      dtype='object')
```

we are going to consider

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

-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
- essyas : Text
- project_titiel : text
```

Assignment 11: TruncatedSVD

- step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title and then find the top 2k words) based on their `idf_` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref (https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/))



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

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

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

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

- step 4 Concatenate these truncatedSVD matrix, with the matrix with features
  - **school_state** : categorical data
  - **clean_categories** : categorical data
  - **clean_subcategories** : categorical data

    - **project_grade_category** :categorical data
    - **teacher_prefix** : categorical data
    - **quantity** : numerical data
    - **teacher_number_of_previously_posted_projects** : numerical data
    - **price** : numerical data
    - **sentiment score's of each of the essay** : numerical data
    - **number of words in the title** : numerical data
    - **number of words in the combine essays** : numerical data
    - **word vectors calculated in** step 3 : numerical data
- step 5: Apply GBDT on matrix that was formed in step 4 of this assignment, **DO REFER THIS BLOG: XGBOOST DMATRIX (https://www.kdnuggets.com/2017/03/simple-xgboost-tutorial-iris-dataset.html)**
- step 6:Hyper parameter tuning (Consider any two hyper parameters)
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

In [131]:

```
pip install xgboost
```

```
Collecting xgboost
  Downloading https://files.pythonhosted.org/packages/5e/49/b95c037b717b4c
eadc76b6e164603471225c27052d1611d5a2e832757945/xgboost-0.90-py2.py3-none-w
in_amd64.whl (https://files.pythonhosted.org/packages/5e/49/b95c037b717b4c
eadc76b6e164603471225c27052d1611d5a2e832757945/xgboost-0.90-py2.py3-none-w
in_amd64.whl) (18.3MB)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.3.1)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.17.1)
Installing collected packages: xgboost
Successfully installed xgboost-0.90
Note: you may need to restart the kernel to use updated packages.
```

In [150]:

```
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_boo

    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
```

2. TruncatedSVD

Testing Co-occurance matrix code before applying it on top 2000 words which are occurred rarely in given corpus

In [242]:

```
import numpy as np
corpus = ["ABC DEF IJK PQR","PQR KLM OPQ","LMN PQR XYZ ABC DEF PQR ABC"]
top_words = ["ABC","PQR","DEF" ]
window_size=2


df = pd.DataFrame()
df['concat_essays_titles'] = corpus
df.head()
```

Out[242]:

	concat_essays_titles
0	ABC DEF IJK PQR
1	PQR KLM OPQ
2	LMN PQR XYZ ABC DEF PQR ABC

In [231]:

```
#
coo_matrix_3x3=(np.zeros((3,3)))
window=2
```

In [232]:

```
for sentance in concat_essays_titles:
    word_sen=sentance.split()
    for idss,word in enumerate(word_sen):
        if word in top_words:
            for i in range(max(0,idss-window),min(idss+window,len(word_sen))):
                if word_sen[i] in top_words:
                    coo_matrix_3x3[top_words.index(word_sen[i]),top_words.index(word)]+
```

In [239]:

```
print(coo_matrix_3x3.shape)
```

```
(3, 3)
```

2.1 Computing Co-occurance matrix

In [182]:

```
coo_matrix=(np.zeros((2000,2000)))
window=5
```

In [183]:

```
final_2000_features=[]
for i in range(2000):
    final_2000_features.append(idf_score_features[i][1])
```

In [184]:

```
for sentance in concat_essays_titles:
    word_sen=sentance.split()
    for idss,word in enumerate(word_sen):
        if word in final_2000_features:
            for i in range(max(0,idss-window),min(idss+window,len(word_sen))):
                if word_sen[i] in final_2000_features:
                    coo_matrix[final_2000_features.index(word_sen[i]),final_2000_featur
```

In [69]:

```
coo_matrix = np.array(coo_matrix)
```

In [235]:

```
coo_matrix
```

Out[235]:

```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 1.]])
```

2.2 Applying TruncatedSVD on co-occurance matrix for dimention reduction

In [71]:

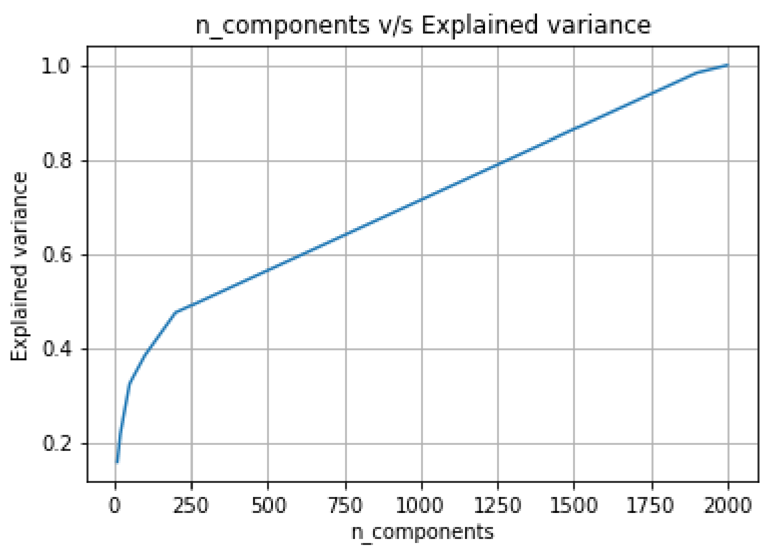
```
#  finding optimal value of n_componenets(n) using truncated svd
from sklearn.decomposition import TruncatedSVD
n_components=[10,20,50,60,100,200,400,500,1000,1200,1500,1700,1800,1900,1999]
explained_variance=[]
for n in n_components:
    svd=TruncatedSVD(n_components=n,random_state=42)
    svd.fit(coo_matrix)
    exvar=svd.explained_variance_ratio_.sum()
    explained_variance.append(exvar)

    print('n_components=',n,'variance=',exvar)
```

```
n_components= 10 variance= 0.15780057219930133
n_components= 20 variance= 0.21804522783821934
n_components= 50 variance= 0.32280942985278505
n_components= 60 variance= 0.3353576384645209
n_components= 100 variance= 0.3835141175728213
n_components= 200 variance= 0.4746408973562684
n_components= 400 variance= 0.5346212639665506
n_components= 500 variance= 0.5645759445391754
n_components= 1000 variance= 0.714353292558117
n_components= 1200 variance= 0.77425139377073
n_components= 1500 variance= 0.8641206516318087
n_components= 1700 variance= 0.9240131577340902
n_components= 1800 variance= 0.9539683514807485
n_components= 1900 variance= 0.9839189220531424
n_components= 1999 variance= 0.9999999999999817
```

2.3 Ploting n_components v/s Explained Variance

In [72]:

```
# ploting curve between n_components and explained_variance
plt.plot(n_components, explained_variance)
plt.xlabel('n_components')
plt.ylabel("Explained variance")
plt.title("n_components v/s Explained variance")
plt.grid()
plt.show()
```



In [73]:

```
from sklearn.decomposition import TruncatedSVD
# got 95 % variance vs explained variance  - 1800
tsvd=TruncatedSVD(n_components=1800,random_state=42)
final_coo_matrix=tsvd.fit_transform(coo_matrix)
```

In [74]:

```
print(final_coo_matrix.shape)
print(final_coo_matrix)
```

```
(2000, 1800)
[[ 8.34928680e-14 -3.76507005e-13 -3.49006635e-13 ...  6.85346944e-03
  -1.19551037e-02 -1.35526925e-02]
 [-4.12573837e-14  2.96351799e-12  2.20875301e-12 ...  3.91722404e-03
  -5.21262643e-03 -1.70517786e-02]
 [-6.30599387e-14 -1.76923458e-12 -1.58643119e-13 ... -2.49691566e-04
  -1.97192405e-02 -1.42059002e-02]
 ...
 [-5.96701889e-14 -1.15225741e-13 -5.63068311e-13 ...  1.05779710e-03
  -7.49475877e-03 -4.70852759e-03]
 [-5.78546607e-18  6.47476970e-16  6.62649109e-16 ... -2.74097268e-16
   2.89361278e-17  1.11030646e-15]
 [-7.59615871e-19 -2.55055037e-17  4.91507179e-17 ...  8.72979916e-16
  -3.49854234e-16 -8.94614803e-16]]
```

2.4 using avg w2v vectorizer method : essays and project_title

In [75]:

```
model = {}
for i in range(len(final_2000_features)):
    model[final_2000_features[i]] = final_coo_matrix[i]
```

In [76]:

```
# model = final_2000_features
glove_words =  set(model.keys())
```

project_title - avg w2v - train

In [80]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_titles_vectors_train = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(X_train['preprocessed_project_title']): # for each review/sentence
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles_vectors_train.append(vector)
```

```
100%|██████████████████████████████████| 49041/49041 [00:00<00:00, 54793.57
it/s]
```

project_title - avg w2v - test

In [81]:

```
avg_w2v_titles_vectors_test = []; # the avg-w2v for each sentence/review is stored in t
for sentence in tqdm(X_test['preprocessed_project_title']): # for each review/sentence
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles_vectors_test.append(vector)
```

```
100%|██████████████████████████████████| 36052/36052 [00:00<00:00, 46368.43
it/s]
```

essays - avg w2v - train

In [84]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essays_vectors_train = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(X_train['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essays_vectors_train.append(vector)
```

```
100%|██████████████████████████████████| 49041/49041 [00:04<00:00, 10501.12
it/s]
```

essays - avg w2v - test

```
In [83]:
```

```
avg_w2v_essays_vectors_test = []; # the avg-w2v for each sentence/review is stored in t
for sentence in tqdm(X_test['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essays_vectors_test.append(vector)
```

```
100%|██████████████████████████████████| 36052/36052 [00:02<00:00, 12829.70
it/s]
```

```
In [99]:
```

```
print("avg w2v essays vector train:",len(avg_w2v_essays_vectors_train))
print("avg w2v essays vector test:",len(avg_w2v_essays_vectors_test))
print("avg w2v titles vector train:",len(avg_w2v_titles_vectors_train))
print("avg w2v titles vector test:",len(avg_w2v_titles_vectors_test))
```

```
avg w2v essays vector train: 49041
avg w2v essays vector test: 36052
avg w2v titles vector train: 49041
avg w2v titles vector test: 36052
```

2.5 word count project_title

In [87]:

```
title_wordcount_train = []
title_train = list(X_train['preprocessed_project_title'])
for i in tqdm(title_train):
    b = len(str(i).split())
    title_wordcount_train.append(b)
title_wordcount_train = np.array(title_wordcount_train)

title_wordcount_test = []
title_test = list(X_test['preprocessed_project_title'])
for i in tqdm(title_test):
    b = len(str(i).split())
    title_wordcount_test.append(b)
title_wordcount_test = np.array(title_wordcount_test)

print(title_wordcount_train.shape)
print(title_wordcount_test.shape)
```

```
100%|████████████████████████████████| 49041/49041 [00:00<00:00, 405274.99
it/s]
100%|████████████████████████████████| 36052/36052 [00:00<00:00, 474338.81
it/s]

(49041,)
(36052,)
```

2.5.1 Standardizing Word counts(TITLES)

In [88]:

```
from sklearn.preprocessing import StandardScaler

title_wordcount_scalar = StandardScaler()
title_wordcount_scalar.fit(title_wordcount_train.reshape(-1,1))

title_wordcount_standardized_train = title_wordcount_scalar.transform(title_wordcount_t
title_wordcount_standardized_test = title_wordcount_scalar.transform(title_wordcount_te

print(title_wordcount_standardized_train.shape)
print(title_wordcount_standardized_test.shape)
```

```
(49041, 1)
(36052, 1)
```

2.6 word count essays

In [91]:

```
essay_wordcount_train = []
essay_train = list(X_train['preprocessed_essays'])
for i in tqdm(essay_train):
    b = len(str(i).split())
    essay_wordcount_train.append(b)
essay_wordcount_train = np.array(essay_wordcount_train)


essay_wordcount_test = []
essay_test = list(X_test['preprocessed_essays'])
for i in tqdm(essay_test):
    b = len(str(i).split())
    essay_wordcount_test.append(b)
essay_wordcount_test = np.array(essay_wordcount_test)

print(essay_wordcount_train.shape)
print(essay_wordcount_test.shape)
```

```
100%|██████████████████████████████████| 49041/49041 [00:00<00:00, 67267.72
it/s]
100%|██████████████████████████████████| 36052/36052 [00:00<00:00, 68666.53
it/s]

(49041,)
(36052,)
```

2.6.1 Standardizing Word counts(ESSAYS)

In [248]:

```
from sklearn.preprocessing import StandardScaler

essay_wordcount_scalar = StandardScaler()
essay_wordcount_scalar.fit(essay_wordcount_train.reshape(-1,1))

essay_wordcount_standardized_train = essay_wordcount_scalar.transform(essay_wordcount_t
essay_wordcount_standardized_test = essay_wordcount_scalar.transform(essay_wordcount_te

print(essay_wordcount_standardized_train.shape)
print(essay_wordcount_standardized_test.shape)
```

```
(49041, 1)
(36052, 1)
```

2.7 Sentiment scores for each essay

In [245]:

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

# link :https://stackoverflow.com/questions/53680690/list-object-has-no-attribute-encod

vader = SentimentIntensityAnalyzer()
essay_train_str = X_train['preprocessed_essays'].str[0].str.join(" ")
sid = essay_train_str.apply(lambda x: vader.polarity_scores(x)['compound'])

sid_train = sid.values.reshape(1,-1).T
print(sid_train)

print("Shape of sid_train and y_train :", sid_train.shape, y_train.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sid_train and y_train : (49041, 1) (49041,)
```

In [246]:

```
# sid test

vader = SentimentIntensityAnalyzer()
essay_test_str = X_test['preprocessed_essays'].str[0].str.join(" ")
sid = essay_test_str.apply(lambda x: vader.polarity_scores(x)['compound'])

sid_test = sid.values.reshape(1,-1).T
print(sid_test)

print("Shape of sid_test and y_test :", sid_test.shape, y_test.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sid_test and y_test : (36052, 1) (36052,)
```

In [247]:

```
# sid cv

vader = SentimentIntensityAnalyzer()
essay_cv_str = X_cv['preprocessed_essays'].str[0].str.join(" ")
sid = essay_cv_str.apply(lambda x: vader.polarity_scores(x)['compound'])
sid_cv = sid.values.reshape(1,-1).T
print(sid_cv)

print("Shape of sid_cv and y_cv :", sid_cv.shape, y_cv.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sid_cv and y_cv : (24155, 1) (24155,)
```

In [243]:

```
print(clean_categories_one_hot_train.shape)
print(clean_subcategories_one_hot_train.shape)
print(school_state_one_hot_train.shape)
print(teacher_prefix_one_hot_train.shape)
print(project_grade_category_one_hot_train.shape)
print(price_normalizer_train.shape)
print(tnppp_normalizer_train.shape)
print(quantity_normalizer_train.shape)
print(essays_tfidf_train.shape)
print(project_title_tfidf_train.shape)
print(len(avg_w2v_essays_vectors_train))
print(len(avg_w2v_titles_vectors_train))
print(len(title_wordcount_standardized_train))
print(len(essay_wordcount_standardized_train))
print(sid_train.shape)
print(final_coo_matrix.shape)
```

```
(49041, 9)
(49041, 30)
(49041, 51)
(49041, 5)
(49041, 4)
(49041, 1)
(49041, 1)
(49041, 1)
(49041, 5000)
(49041, 2111)
49041
49041
49041
49041
(49041, 1)
(2000, 1800)
```

2.8 Merge matrices

In [122]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
set_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
               teacher_prefix_one_hot_train, project_grade_category_one_hot_train,
                price_normalizer_train, tnppp_normalizer_train,quantity_normalizer_trai
               title_wordcount_standardized_train,essay_wordcount_standardized_train,
               avg_w2v_essays_vectors_train, avg_w2v_titles_vectors_train)).tocsr()

set_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sch
               teacher_prefix_one_hot_test, project_grade_category_one_hot_test,
               price_normalizer_test, tnppp_normalizer_test, quantity_normalizer_test,
               sid_test, title_wordcount_standardized_test,essay_wordcount_standardized_
               avg_w2v_essays_vectors_test, avg_w2v_titles_vectors_test)).tocsr()

print("Final merged datamatrix of set_train:",set_train.shape, y_train.shape)
print("Final merged datamatrix of set_test: ",set_test.shape, y_test.shape)
print("="*100)
```

```
Final merged datamatrix of set_train: (49041, 3705) (49041,)
Final merged datamatrix of set_test:  (36052, 3705) (36052,)
==========================================================================
==========================
```

2.9 Apply XGBoost on the Final Features from the above section

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

In [140]:

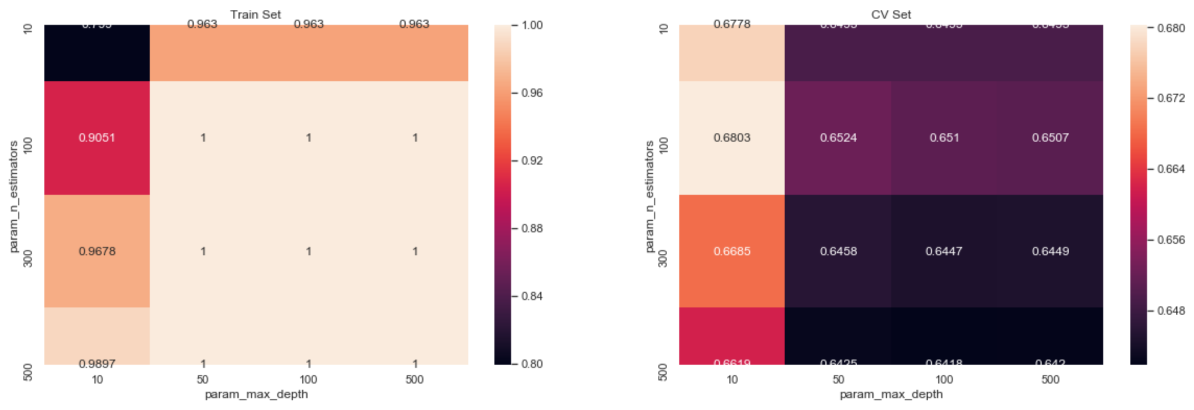
```
%%time
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt
import xgboost as xgb
import time

start_time = time.time()
gbdt = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 300, 500], "learning_rate":[ 0.01, 0.1 , 1]}
clf1 = GridSearchCV(gbdt, parameters, cv= 3, scoring='roc_auc',return_train_score=True)
clf1.fit(set_train, y_train)
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 23555.665098667145 ms
Wall time: 6h 32min 35s
```

2.9.1Hyper parameter Tuning to find best estimator :: GridSearchCV set

In [142]:

```
# ploting the performance of model both on train data and cross validation data for eac
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(clf1.cv_results_).groupby(['param_n_estimators', 'param_lear

fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [144]:

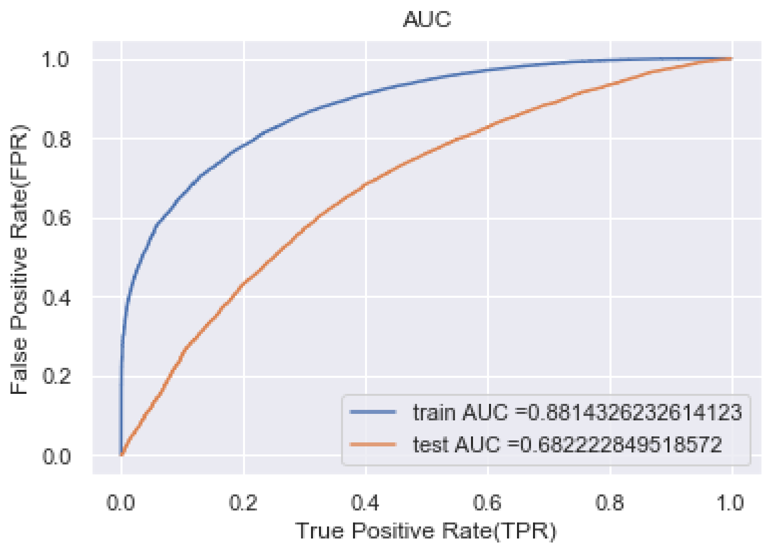
```
# Best tune parameters with score
print(clf1.best_estimator_)
print(clf1.score(set_train,y_train))
print(clf1.score(set_test,y_test))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=10,
              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)
0.8814326232614123
0.682222849518572
```

2.9.2 Train model using the best hyperparameter

In [145]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_1v =xgb.XGBClassifier (class_weight = 'balanced',learning_rate=10,n_estimator=100)
clf_1v.fit(set_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_1v.predict_proba(set_train)[:,
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_1v.predict_proba(set_test)[:,1])

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.grid(True)
plt.xlabel("False Positive Rate(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.show()
```



```
Wall time: 5min 3s
```

2.9.3 Confustion Matrix : set_train and set_test

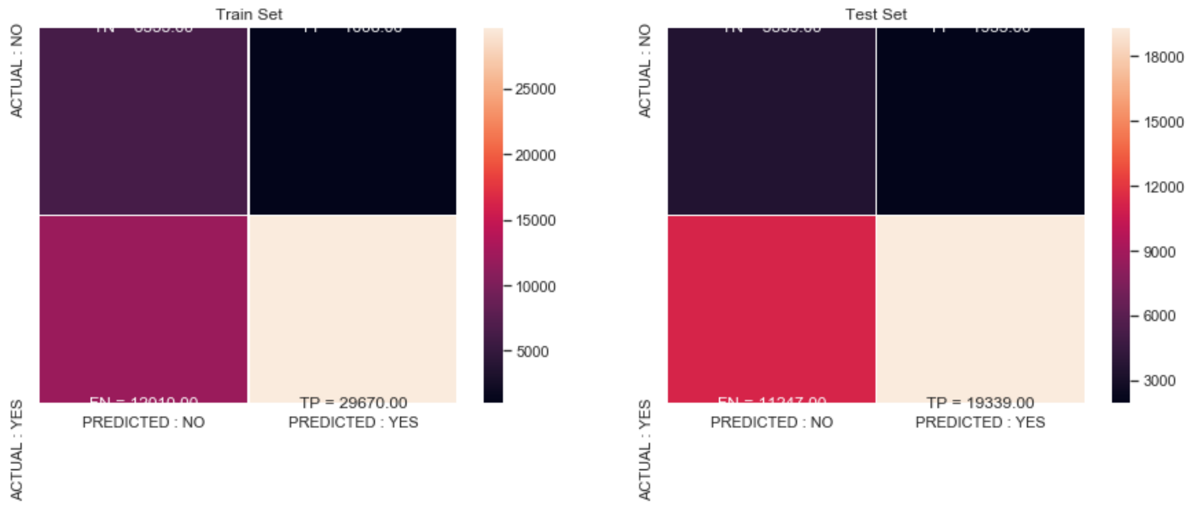
In [146]:

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

In [147]:

```
y_train_pred_1 = clf_1v.predict_proba(set_train)[:,1]
y_test_pred_1 = clf_1v.predict_proba(set_test)[:,1]


#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_1 = confusion_matrix(y_train, predict(y_train_pred_1, thresholds, tr
conf_matr_df_test_1 = confusion_matrix(y_test, predict(y_test_pred_1, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

sns.heatmap(conf_matr_df_train_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.6258925691530375 for threshold 0.851
the maximum value of tpr*(1-fpr) 0.41074889661678426 for threshold 0.844
```



3. Conclusion

In [148]:

```
from prettytable import PrettyTable
p = PrettyTable()
p.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]
p.add_row(["AVG W2V", "XGBoost", "Max Depth:10 , n_estimators:100", 0.68])
print(p)
```

```
+------------+---------+---------------------------------+------+
| Vectorizer |  Model  |         Hyper Parameter         | AUC  |
+------------+---------+---------------------------------+------+
|  AVG W2V   | XGBoost | Max Depth:10 , n_estimators:100 | 0.68 |
+------------+---------+---------------------------------+------+
```

Thank You.

Sign Off Ramesh Battu (https://www.linkedin.com/in/rameshbattuai/)