DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
```

```
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

from sklearn.metrics import accuracy_score
```

1.1 Reading Data

In [2]:

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

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
```

```
Number of data points in train data (50000, 17)
--------------------------------------------------
```

In [4]:

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


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


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


project_data.head(2)
```

Out[4]:

	Unnamed:	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_s

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_s
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Grades PreK-2	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5	L

In [5]:

```
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[5]:

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

In [6]:

```
project_grade_category = []
for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" ", "_")
    project_grade_category.append(a)
```

In [7]:

```
project_grade_category[0:5]
```

Out[7]:

```
['Grades_PreK-2', 'Grades_6-8', 'Grades_6-8', 'Grades_PreK-2', 'Grades_PreK-2']
```

In [8]:

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

In [9]:

```
project_data["project_grade_category"] = project_grade_category
```

In [10]:

```
project_data.head(5)
```

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_subject_categories	proje
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Applied Learning	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Literacy & Language	

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_subject_categories	proje
23374	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	Ms.	CA	2016-04-27 02:04:15	Literacy & Language	
49228	57854	p099430	4000cfe0c8b2df75a218347c1765e283	Ms.	IL	2016-04-27 07:19:44	Literacy & Language	

1.2 preprocessing of `project_subject_categories`

In [11]:

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

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [12]:

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

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i
```

```
Science -> Math&Science
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

In [13]:

```
#Introducing new coloumn number_of_Words in Title
word_count_title = []
for a in project_data["project_title"] :
    b = len(a.split())
    word_count_title.append(b)
project_data["title_word_count"] = word_count_title
project_data.head(5)
```

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	project_essay_1	pr
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	W
29891	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	Mrs.	CA	2016-04-27 01:10:09	Breakout Box to Ignite Engagement!	It's the end of the school year. Routines have...	
23374	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	Ms.	CA	2016-04-27 02:04:15	iPad for Learners	Never has society so rapidly changed. Technolo...	
49228	57854	p099430	4000cfe0c8b2df75a218347c1765e283	Ms.	IL	2016-04-27 07:19:44	A flexible classroom for flexible minds!	My students yearn for a classroom environment ...	

1.3 Text preprocessing

In [14]:

```
# 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)
```

Adding "essay_word_count " feature into project_data dataset

In [15]:

```
#Adding essay_word_count feature into project_data dataset
essay_word_count = []
```

```
    essay_word_count.append(c)
project_data["essay_word_count"] = essay_word_count
project_data.head(5)
```

Out[15]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	project_essay_1	pr
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...	W
29891	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	Mrs.	CA	2016-04-27 01:10:09	Breakout Box to Ignite Engagement!	It's the end of the school year. Routines have...	
23374	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	Ms.	CA	2016-04-27 02:04:15	iPad for Learners	Never has society so rapidly changed. Technolo...	
49228	57854	p099430	4000cfe0c8b2df75a218347c1765e283	Ms.	IL	2016-04-27 07:19:44	A flexible classroom for flexible minds!	My students yearn for a classroom environment ...	

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

In [16]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [17]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data,
project_data['project_is_approved'], test_size=0.33, stratify = project_data['project_is_approved'
])
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

In [18]:

```
X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

In [19]:

```
print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[500])
print("="*50)
```

```
print(X_train['essay'].values[10000])
print("="*50)
print(X_train['essay'].values[20000])
print("="*50)
```

```
We are a school with Mandarin immersion program. My class is consists of 24 kindergarteners (age 5
) who spend 80% of their school time learning in Mandarin and 20% in English. Almost all of the st
udents come from highly educated families.  \r\n\r\nMy kindergarteners are academically high-
performing, but they also LOVE hands-on activities. Instead of greeting them with a worksheet ever
y morning, my goals is to have them engaged in a hands on activity.  I believe that young children
can benefit a lot from playing, creating, building, and imagination.These hands on activities can
benefit my kindergarteners in many different ways.  They encourage the little one's imagination an
d creativity.  In the world of imagination, there's no right or wrong answers. Therefore, those
who are extremely self-conscious will have the courage to take risk and try out new
creations.\r\n\r\nWith these hands on activity sets, my kindergarteners not only will improve
their fine motor skills, but also their hand eye coordination.   These skills are essential for le
arners at a young age. More importantly, the kindergartners will learn to share the activity sets
with their classmates and thus they will learn to work cooperatively.nannan
==================================================
I teach a wonderful group of 23 second graders. They come to school excited to start the day and r
eady to learn. Our school is located in a rural farming community, and the building houses kinderg
arten through sixth grade with four classroom at each grade level. My students come from homes
whose parents work in the farming, coal mining, and health industries, just to name a few. We have
a very supportive parent system and our school is growing technologically.Adding seven Chromebooks
to our classroom will be a huge asset to the students. They will allow us to have easier access to
I-Ready, Accelerated Reader, Spelling City, and would make it much simpler to research various top
ics without leaving the classroom. They will also aide in supplementing classroom instruction.
\r\nStudents will be able to utilize all Google programs to assist them in their learning and unde
rstanding of Language Arts and Math topics. Classroom instruction time will be utilized more effic
iently because students will not need to leave the classroom as often for technology usage. Our gr
ade level goal is to accumulate enough Chromebooks to support a whole classroom.nannan
==================================================
My students are in third grade and come from all backgrounds, which makes them all unique and a pl
easure to teach. They bring excitement to the classroom and are eager to learn about all kinds of
subjects that relate to the real world. \r\n\r\nMy students are willing to work hard to meet their
goals and I want them to feel supported and have all the resources possible to achieve success bot
h in and out of the classroom. They love to read and discover new information by researching and w
orking together as a team. \r\n\r\nOur school is 100% free lunch and many of the students face set
backs, but that does not stop them from wanting to come to school to learn, succeed, and feel love
d everyday!   \r\nThird grade students would benefit from a communication center in our classroom
to keep their materials, such as daily classwork, homework, library books, and papers to be sent h
ome, organized and in their proper place. \r\n\r\nA classroom communication center would teach stu
dents how to be responsible and stay organized so they can focus on learning!\r\n\r\nStudent's wil
l have a place for everything, and everything in it's place so our third grade classroom can be as
functional and productive as possible. Organization is an important tool in a student's life.
Students will be able to focus on learning by knowing where to go for materials and
resources.nannan
==================================================
As a teacher in a small school district, my students are faced with challenges both in and out of
our classroom. Though faced with many challenges, I am working to keep things simple, yet fun. I a
im to provide my students with creative and meaningful learning experiences each and every day.\r\
n\r\nMy students are creative, clever and very spontaneous.\r\nEach and every one is unique in the
ir own way. In our classroom we love to move, read and receive positive reinforcement. Many of the
m are being raised in single parent households and receive a free lunch based on their
socioeconomic status. These things may prevent them from getting ahead early in life and may not p
rovide them with the life experiences many of us see as \"typical.\"  From the minute they walk in
to our classroom I focus on their potential and growth while they are with me. Though I may not be
able to control their lives outside of our classroom, I can certainly control their experiences th
roughout the school day. By doing this in a creative and positive way, I am hopeful to inspire stu
dents to continue on an enriched path.Before the 2016-17 school year, my class was funded for a li
stening center and books for kids to follow along with. This has been a great tool in my
classroom, however the project wouldn't be complete without a designated area in the classroom (ne
ar our class library) for kids to cozy up and read.\r\n     This project will help complete the li
stening center project we have been working towards. I am looking forward to complete this center
and having an area for my kids to go to that is relaxed and organized for them to read and listen
to some of their favorite stories!nannan
==================================================
My students face various challenges in their communities daily. 100% of my students qualify for fr
ee/reduced lunch and some of their parents have a difficult time finding the resources to provide
them with basic school supplies. They still show up to school looking for safety, consistency, att
ention, and love.\r\nThese kids are growing up seeing much violence in their community, are pressu
red to join gangs, try drugs, or get involved in other illegal activities that will negatively imp
act their lives in the long run. \r\nOur students are driven, talented, and exceptional. They OOZE
with compassion and leadership! They are beginning a Student Leadership Club (Student Council) and
```

e teachers do not only teach reading, writing, and math. We teach our kids how to be kind to each other, how to advocate for their own needs, and how to work together even if you don't agree with someone else's ideas. We show our students they can be successful no matter what challenges they m ay face. \r\nSadly, the students at Roosevelt have not had an active Student Council in over 3 yea rs. I decided to take the initiative and bring the student government back to life! Students applied and collected recommendations proving to be young, ambitious leaders.\r\n\r\nMy students h ave goals to host a canned food drive, a school dance and complete several various community outre ach events this school year. \r\n\r\nMembers are 5th through 8th graders interested in planning, o rganizing and supporting school and community activities and events. The materials they have now a re very limited and they are in dire need of new and additional materials to assist advertising an d executing these events.nannan
==

Split dataset into train and test datasets

In [20]:

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

In [21]:

```
# 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[20000])
print(sent)
print("="*50)
```

I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school pr ovides free breakfast for all students. I am a Special Education certified teacher and I teach Kin dergarten in a general education setting with my class that consists 52% students with special nee ds. The disabilities include Autism Spectrum Disorder, Speech Impaired, Language Impaired, Other H ealth Impaired (ADHD), and Developmentally Delayed. I also have about 42% of my students who are E nglish Language Learners.\r\n\r\n\"Self-motivated learners\" is a synonym of \"my students\". The y love to learn and they possess a positive outlook and attitude in school. Almost everyday, my st udents would ask me, \"Ms. Perez, what are we going to learn today?\" I could not ask for a better greeting from my students.This project will greatly impact my students' learning on a daily basis. The wobble chairs will provide assistance for my students who have difficulties focusing and atten ding during lessons and discussions. Despite the fact that students participate in physical activi ties in P.E., Recess, and GoNoodle (dance videos) sessions in our classroom, students still have e nergy to stand or wiggle from their seats during lessons. Due to these special needs that are beyond the students' control, there is a lot of distraction and student learning is not really ach ieved at its full potential. The lack of appropriate stimulation hinders them to focus and learn i n class. Students with special needs will be able to sit on the wobble chairs during whole group/small group lessons. This will enable their little active bodies to move while "sitting stil l" without disrupting other students. As a result, all students will improve focus and increase st udent attention in learning all content areas. In addition, the visual timer will help my students to actually see the allotted time for activities. This will benefit especially ELL students and st udents with special needs. Whenever we do independent classwork or work in our centers, the studen ts can refer to it and self-monitor their progress in completing assignments. It will encourage th em to use their time wisely and finish tasks on time. It will also help the students have a smooth er transition from one activity to another. \r\nBy donating to this project, you will

me set all students for success! I am looking forward to seeing my students become active listener
s and engaged learners, and always happy to go to school!\r\nnannan
==

In [23]:

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

I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school pr
ovides free breakfast for all students. I am a Special Education certified teacher and I teach Kin
dergarten in a general education setting with my class that consists 52% students with special nee
ds. The disabilities include Autism Spectrum Disorder, Speech Impaired, Language Impaired, Other H
ealth Impaired (ADHD), and Developmentally Delayed. I also have about 42% of my students who are E
nglish Language Learners. Self-motivated learners is a synonym of my students . They love
to learn and they possess a positive outlook and attitude in school. Almost everyday, my students
would ask me, Ms. Perez, what are we going to learn today? I could not ask for a better greeting
from my students.This project will greatly impact my students' learning on a daily basis. The wobb
le chairs will provide assistance for my students who have difficulties focusing and attending dur
ing lessons and discussions. Despite the fact that students participate in physical activities in
P.E., Recess, and GoNoodle (dance videos) sessions in our classroom, students still have energy to
stand or wiggle from their seats during lessons. Due to these special needs that are beyond the st
udents' control, there is a lot of distraction and student learning is not really achieved at its
full potential. The lack of appropriate stimulation hinders them to focus and learn in class. Stud
ents with special needs will be able to sit on the wobble chairs during whole group/small group le
ssons. This will enable their little active bodies to move while "sitting still" without
disrupting other students. As a result, all students will improve focus and increase student atten
tion in learning all content areas. In addition, the visual timer will help my students to actuall
y see the allotted time for activities. This will benefit especially ELL students and students wit
h special needs. Whenever we do independent classwork or work in our centers, the students can ref
er to it and self-monitor their progress in completing assignments. It will encourage them to use
their time wisely and finish tasks on time. It will also help the students have a smoother transit
ion from one activity to another. By donating to this project, you will significantly help stude
nts with special needs have an equal opportunity to learn with their peers. Behavior issues will b
e greatly minimized and classroom management will be optimized. Help me set all students for succe
ss! I am looking forward to seeing my students become active listeners and engaged learners, and a
lways happy to go to school! nannan

In [24]:

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

I teach at a Title 1 school with 73 of my students who receive free reduced lunch Our school
provides free breakfast for all students I am a Special Education certified teacher and I teach Ki
ndergarten in a general education setting with my class that consists 52 students with special nee
ds The disabilities include Autism Spectrum Disorder Speech Impaired Language Impaired Other Healt
h Impaired ADHD and Developmentally Delayed I also have about 42 of my students who are English La
nguage Learners Self motivated learners is a synonym of my students They love to learn and they po
ssess a positive outlook and attitude in school Almost everyday my students would ask me Ms Perez
what are we going to learn today I could not ask for a better greeting from my students This proje
ct will greatly impact my students learning on a daily basis The wobble chairs will provide assist
ance for my students who have difficulties focusing and attending during lessons and discussions D
espite the fact that students participate in physical activities in P E Recess and GoNoodle dance
videos sessions in our classroom students still have energy to stand or wiggle from their seats du
ring lessons Due to these special needs that are beyond the students control there is a lot of dis
traction and student learning is not really achieved at its full potential The lack of appropriate
stimulation hinders them to focus and learn in class Students with special needs will be able to
sit on the wobble chairs during whole group small group lessons This will enable their little
active bodies to move while sitting still without disrupting other students As a result all
students will improve focus and increase student attention in learning all content areas In
addition the visual timer will help my students to actually see the allotted time for activities T
his will benefit especially ELL students and students with special needs Whenever we do
independent classwork or work in our centers the students can refer to it and self monitor their p
rogress in completing assignments It will encourage them to use their time wisely and finish tasks
on time It will also help the students have a smoother transition from one activity to another By
donating to this project you will significantly help students with special needs have an equal

y students become active listeners and engaged learners and always happy to go to school nannan

In [25]:

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

In [26]:

```
#Train data preprocessing(text)

#combining all train data into preprocessed_essays_train
from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentence in tqdm(X_train['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_train.append(sent.lower().strip())
```

```
100%|| 22445/22445 [00:
38<00:00, 585.58it/s]
```

In [27]:

```
# after preprocesing
preprocessed_essays_train[1000]
```

Out[27]:

```
'students third grade come backgrounds makes unique pleasure teach bring excitement classroom
eager learn kinds subjects relate real world students willing work hard meet goals want feel
supported resources possible achieve success classroom love read discover new information research
ing working together team school 100 free lunch many students face setbacks not stop wanting come
school learn succeed feel loved everyday third grade students would benefit communication center
classroom keep materials daily classwork homework library books papers sent home organized proper
place classroom communication center would teach students responsible stay organized focus
learning student place everything everything place third grade classroom functional productive poss
```

In [28]:

```
#Test data preprocessing(textual)

preprocessed_essays_test = []
# tqdm is for printing the status bar
for sentence in tqdm(X_test['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_test.append(sent.lower().strip())
```

```
100%|| 16500/16500 [00:23<00:00, 688.76it/s]
```

In [29]:

```
# after preprocesing
preprocessed_essays_test[1000]
```

Out[29]:

```
'child cannot learn way teach maybe teach way learn ignacio estrada classroom students asked apply conceptual understanding mathematics create models solve real life application problems students come low socio economic community high population migrant english language learners students hard working dedicated education school provides students multiple ways learn material grade level classroom garden classes students use materials requested math intervention course math stations within core math classroom middle school students still struggle conceptually fractions patterns multiplication great disadvantage peers materials go back basics teach concepts foundation level reinforce skills project make difference provide students struggled math whole life second chance learn concepts ground time get students 8th grade many already developed love hatred math resources opportunity help many students change perspectives better close gap low high performing students'
```

In [30]:

```
#Cross validation data preprocessing (text)

preprocessed_essays_cv = []
# tqdm is for printing the status bar
for sentence in tqdm(X_cv['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_cv.append(sent.lower().strip())
```

```
100%|| 11055/11055 [00:15<00:00, 702.35it/s]
```

In [31]:

```
# data after preprocesing
preprocessed_essays_cv[1000]
```

Out[31]:

```
'wow much say students class students special even though hardest hurdles jump survive come school positive attitudes excitement learning always seem try best even though things get extremely difficult glow faces everything clicks proven learned something one favorite looks time materials impact students allowing take control learning work independently master skills need move next grade 3rd one big things struggle teacher creating effective math centers already made matching standards
```

1.4 Preprocessing of `project_title`

In [32]:

```
# similarly you can preprocess the titles also
# printing some random essays.
print(project_data['project_title'].values[2000])
print("="*50)
print(project_data['project_title'].values[1500])
print("="*50)
print(project_data['project_title'].values[10000])
print("="*50)
print(project_data['project_title'].values[12000])
print("="*50)
print(project_data['project_title'].values[1699])
print("="*50)
```

```
Boards to Stop the Boredom!
==================================================
6th grade authors will change The WORLD one letter at a tim
==================================================
AVID Students Success: Basic Supplies for Organization Skills
==================================================
Wiggle, Wiggle, Wiggle: Let Them Move So They Can Learn.
==================================================
Making Math Come Alive for Students with Special Needs!
==================================================
```

In [33]:

```
#Preprocessing the Project Title coloumn in Train data

preprocessed_titles_train = []

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

```
100%|| 22445/22445 [00:01<00:00, 15452.36it/s]
```

In [34]:

```
#after preprocessing the project title coloumn in train data
preprocessed_titles_train[1000]
```

Out[34]:

```
'communication key'
```

Preprocessing of project title for test data

In [35]:

```
#Preprocessing project title for test data

preprocessed_titles_test = []

for titles in tqdm(X_test["project_title"]):
    title = decontracted(titles)
```

```
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_titles_test.append(title.lower().strip())
```

```
100%|| 16500/16500 [00:01<00:00, 15570.98it/s]
```

In [36]:

```
#after preprocessing the project title coloumn in train data

preprocessed_titles_test[1000]
```

Out[36]:

```
'all hands math'
```

In [37]:

```
#Preprocessing project title in cross validation data

preprocessed_titles_cv = []

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

```
100%|| 11055/11055 [00:00<00:00, 13317.66it/s]
```

In [38]:

```
#after preprocessing the project title coloumn in cross validation data


preprocessed_titles_cv[1000]
```

Out[38]:

```
'independent learning materials'
```

1.5 Preparing data for models

In [39]:

```
project_data.columns
```

Out[39]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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',
       'project_grade_category', 'clean_categories', 'clean_subcategories',
       'title_word_count', 'essay', 'essay_word_count'],
      dtype='object')
```

we are going to consider

- school_state : categorical data

- project_grade_category : categorical data
- teacher_prefix : categorical data

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

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

1.5.1 Vectorizing Categorical data

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

2.2 Make Data Model Ready: encoding categorical and numerical data

In [41]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

In [40]:

```
# we use count vectorizer to convert the values into one

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
)
vectorizer.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer.transform(X_train['clean_categories'].values)
categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].values)
categories_one_hot_cv = vectorizer.transform(X_cv['clean_categories'].values)

print(vectorizer.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",categories_one_hot_test.shape)
print("Shape of matrix of CV data after one hot encoding ",categories_one_hot_cv.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix of Train data after one hot encoding  (22445, 9)
Shape of matrix of Test data after one hot encoding  (16500, 9)
Shape of matrix of CV data after one hot encoding  (11055, 9)
```

In [41]:

```
#we use count vectorizer to convert the values into one

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

sub_categories_one_hot_train = vectorizer.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)


print(vectorizer.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",sub_categories_one_hot_train.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix of Train data after one hot encoding  (22445, 30)
Shape of matrix of Test data after one hot encoding  (16500, 30)
Shape of matrix of Cross Validation data after one hot encoding  (11055, 30)
```

In [42]:

```
#school_state one hot encoding
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [43]:

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

In [44]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()), lowercase=False
, binary=True)
vectorizer.fit(X_train['school_state'].values)

school_state_categories_one_hot_train = vectorizer.transform(X_train['school_state'].values)
school_state_categories_one_hot_test = vectorizer.transform(X_test['school_state'].values)
school_state_categories_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)

print(vectorizer.get_feature_names())

print("Shape of matrix of Train data after one hot encoding
",school_state_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_categories_one_hot_test.
shape)
print("Shape of matrix of Cross Validation data after one hot encoding
",school_state_categories_one_hot_cv.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'NH', 'SD', 'NE', 'AK', 'DE', 'WV', 'ME', 'NM', 'HI', 'DC', 'KS', 'I
D', 'IA', 'AR', 'CO', 'MN', 'OR', 'MS', 'KY', 'NV', 'MD', 'TN', 'CT', 'AL', 'UT', 'WI', 'VA', 'AZ',
'NJ', 'OK', 'MA', 'LA', 'WA', 'MO', 'IN', 'OH', 'PA', 'MI', 'GA', 'SC', 'IL', 'NC', 'FL', 'TX', 'NY
', 'CA']
Shape of matrix of Train data after one hot encoding  (22445, 51)
Shape of matrix of Test data after one hot encoding  (16500, 51)
Shape of matrix of Cross Validation data after one hot encoding  (11055, 51)
```

In [45]:

```
#performing One Hot Encoding on Project Grade Category

my_counter = Counter()
for project_grade in project_data['project_grade_category'].values:
    my_counter.update(project_grade.split())
```

In [46]:

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

In [47]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), lowercase=Fals
e, binary=True)
vectorizer.fit(X_train['project_grade_category'].values)

project_grade_categories_one_hot_train = vectorizer.transform(X_train['project_grade_category'].va
lues)
project_grade_categories_one_hot_test =
vectorizer.transform(X_test['project_grade_category'].values)
project_grade_categories_one_hot_cv = vectorizer.transform(X_cv['project_grade_category'].values)

print(vectorizer.get_feature_names())

print("Shape of matrix of Train data after one hot encoding
",project_grade_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_one_hot_test
.shape)
print("Shape of matrix of Cross Validation data after one hot encoding
",project_grade_categories_one_hot_cv.shape)
```

```
['Grades_9-12', 'Grades_6-8', 'Grades_3-5', 'Grades_PreK-2']
Shape of matrix of Train data after one hot encoding  (22445, 4)
Shape of matrix of Test data after one hot encoding  (16500, 4)
Shape of matrix of Cross Validation data after one hot encoding  (11055, 4)
```

In [48]:

```
#performing One Hot Encoding on Teacher prefix Category

my_counter = Counter()
for teacher_prefix in project_data['teacher_prefix'].values:
    teacher_prefix = str(teacher_prefix)
    my_counter.update(teacher_prefix.split())
```

In [49]:

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

In [50]:

```
## we use count vectorizer to convert the values into one hot encoded features
## Unlike the previous Categories this category returns a
## ValueError: np.nan is an invalid document, expected byte or unicode string.
## The link below explains h0w to tackle such discrepancies.
## https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-
is-an-invalid-document/39308809#39308809

vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys()), lowercase=Fal
se, binary=True)
vectorizer.fit(X_train['teacher_prefix'].values.astype("U"))

teacher_prefix_categories_one_hot_train =
vectorizer.transform(X_train['teacher_prefix'].values.astype("U"))
teacher_prefix_categories_one_hot_test =
vectorizer.transform(X_test['teacher_prefix'].values.astype("U"))
teacher_prefix_categories_one_hot_cv =
vectorizer.transform(X_cv['teacher_prefix'].values.astype("U"))

print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_test.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_cv.shape)
```

```
['nan', 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encoding  (22445, 6)
Shape of matrix after one hot encoding  (16500, 6)
Shape of matrix after one hot encoding  (11055, 6)
```

Make Data Model Ready: encoding essay, and project_title

In [51]:

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

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

Bag of words

In [52]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).

vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(preprocessed_essays_train)

text_bow_train = vectorizer.transform(preprocessed_essays_train)

print("Shape of matrix after one hot encoding ",text_bow_train.shape)
```

```
Shape of matrix after one hot encoding  (22445, 8733)
```

In [53]:

```
text_bow_test = vectorizer.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (16500, 8733)
```

In [54]:

```
text_bow_cv = vectorizer.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encoding ",text_bow_cv.shape)
```

```
Shape of matrix after one hot encoding  (11055, 8733)
```

In [55]:

```
vectorizer.fit(preprocessed_titles_train)
title_bow_train = vectorizer.transform(preprocessed_titles_train)
print("Shape of matrix after one hot encoding ",title_bow_train.shape)
```

```
Shape of matrix after one hot encoding  (22445, 1225)
```

In [56]:

```
title_bow_test = vectorizer.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (16500, 1225)
```

In [57]:

```
print("Shape of matrix after one hot encoding ",title_bow_cv.shape)
```

```
Shape of matrix after one hot encoding  (11055, 1225)
```

1.5.2.2 TFIDF vectorizer

In [58]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(preprocessed_essays_train)

text_tfidf_train = vectorizer.transform(preprocessed_essays_train)
print("Shape of matrix after one hot encoding ",text_tfidf_train.shape)
```

```
Shape of matrix after one hot encoding  (22445, 8733)
```

In [59]:

```
text_tfidf_test = vectorizer.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (16500, 8733)
```

In [60]:

```
text_tfidf_cv = vectorizer.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encoding ",text_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (11055, 8733)
```

In [61]:

```
vectorizer = TfidfVectorizer(min_df=10)

vectorizer.fit(preprocessed_titles_train)
title_tfidf_train = vectorizer.transform(preprocessed_titles_train)
print("Shape of matrix after one hot encoding ",title_tfidf_train.shape)
```

```
Shape of matrix after one hot encoding  (22445, 1225)
```

In [62]:

```
title_tfidf_test = vectorizer.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (16500, 1225)
```

In [63]:

```
title_tfidf_cv = vectorizer.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encoding ",title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (11055, 1225)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [64]:

```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
```

```
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
```

In [65]:

```
model = loadGloveModel('glove.42B.300d.txt')
```

```
Loading Glove Model

1917495it [20:57, 1524.76it/s]

Done. 1917495  words loaded!
```

In [66]:

```
words_train_essays = []

for i in preprocessed_essays_train :
    words_train_essays.extend(i.split(' '))

## Find the total number of words in the Train data of Essays.

print("all the words in the corpus", len(words_train_essays))
## Find the unique words in this set of words

words_train_essay = set(words_train_essays)
print("the unique words in the corpus", len(words_train_essay))

## Find the words present in both Glove Vectors as well as our corpus.

inter_words = set(model.keys()).intersection(words_train_essay)

print("The number of words that are present in both glove vectors and our corpus are {} which \
is nearly {}% ".format(len(inter_words), np.round((float(len(inter_words))/len(words_train_essay))
*100)))

words_corpus_train_essay = {}

words_glove = set(model.keys())

for i in words_train_essay:
    if i in words_glove:
        words_corpus_train_essay[i] = model[i]

print("word 2 vec length", len(words_corpus_train_essay))

# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-and-load-variables-in-python/

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


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

```
all the words in the corpus 3090886
the unique words in the corpus 30279
The number of words that are present in both glove vectors and our corpus are 28617 which is
```

Train-Essays

In [67]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_train = [];

for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

```
100%|| 22445/22445 [00:16<00:00, 1397.76it/s]
```

```
22445
300
```

Test-Essays

In [68]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_test = [];

for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
100%| 16500/16500 [00:42<00:00, 391.64it/s]
```

```
16500
300
```

Cross-validation essays

In [70]:

```
# average Word2Vec
```

```
avg_w2v_vectors_cv = [];

for sentence in tqdm(preprocessed_essays_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))
```

```
100%|██████████| 11055/11055
[00:07<00:00, 1526.31it/s]
```

```
11055
300
```

1.4.2.6 Using Pretrained Models: AVG W2V on `project_title`

Train titles

In [71]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_train.append(vector)

print(len(avg_w2v_vectors_titles_train))
print(len(avg_w2v_vectors_titles_train[0]))
```

```
100%|██████████| 22445/22445
[00:01<00:00, 16414.30it/s]
```

```
22445
300
```

Test titles

In [72]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
```

```
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_test.append(vector)

print(len(avg_w2v_vectors_titles_test))
print(len(avg_w2v_vectors_titles_test[0]))
```

```
100%|██████████| 16500/16500
[00:02<00:00, 8050.68it/s]
```

```
16500
300
```

Cross validation titles

In [73]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_cv.append(vector)

print(len(avg_w2v_vectors_titles_cv))
print(len(avg_w2v_vectors_titles_cv[0]))
```

```
100%|██████████| 11055/11055
[00:00<00:00, 16908.05it/s]
```

```
11055
300
```

Using Pretrained Models: TFIDF weighted W2V

Train essays

In [74]:

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

In [75]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
```

```
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))
```

```
100%|| 22445/22445 [01:
41<00:00, 220.93it/s]
```

```
22445
300
```

In [76]:

```
# Similarly you can vectorize for title also
```

Test essays

In [77]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
100%|| 16500/16500 [01:
07<00:00, 243.36it/s]
```

```
16500
300
```

Cross validation essays

In [78]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
```

```
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

```
100%|██████████| 11055/11055 [00:
53<00:00, 207.43it/s]
```

```
11055
300
```

Train essays

In [79]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [80]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_train = [];

for sentence in tqdm(preprocessed_titles_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_train.append(vector)

print(len(tfidf_w2v_vectors_titles_train))
print(len(tfidf_w2v_vectors_titles_train[0]))
```

```
100%|██████████| 22445/22445
[00:01<00:00, 12891.61it/s]
```

```
22445
300
```

Test essays

In [81]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_test = [];
```

```
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_test.append(vector)

print(len(tfidf_w2v_vectors_titles_test))
print(len(tfidf_w2v_vectors_titles_test[0]))
```

```
100%|██████████| 16500/16500
[00:01<00:00, 14877.50it/s]
```

```
16500
300
```

Cross validation essays

In [82]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_cv = [];

for sentence in tqdm(preprocessed_titles_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_cv.append(vector)

print(len(tfidf_w2v_vectors_titles_cv))
print(len(tfidf_w2v_vectors_titles_cv[0]))
```

```
100%|██████████| 11055/11055
[00:00<00:00, 12249.54it/s]
```

```
11055
300
```

Vectorizing Numerical features

price

In [83]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in
-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
```

Out[83]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [84]:

```
# join two dataframes in python:
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
```

In [85]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

normalizer.fit(X_train['price'].values.reshape(-1,1))

price_train = normalizer.transform(X_train['price'].values.reshape(-1,1))
price_cv = normalizer.transform(X_cv['price'].values.reshape(-1,1))
price_test = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
print("="*100)
```

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

Quantity

In [86]:

```
normalizer = Normalizer()

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

normalizer.fit(X_train['quantity'].values.reshape(-1,1))

quantity_train = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(quantity_train.shape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
```

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

No.of projects posted previously by teachers

In [87]:

```
normalizer = Normalizer()

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

normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

prev_projects_train = normalizer.transform(X_train['teacher_number_of_previously_posted_projects']
.values.reshape(-1,1))
prev_projects_cv =
normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].v
alues.reshape(-1,1))

print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
print("="*100)
```

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

Title_word_count

In [88]:

```
normalizer = Normalizer()

normalizer.fit(X_train['title_word_count'].values.reshape(-1,1))

title_word_count_train = normalizer.transform(X_train['title_word_count'].values.reshape(-1,1))
title_word_count_cv = normalizer.transform(X_cv['title_word_count'].values.reshape(-1,1))
title_word_count_test = normalizer.transform(X_test['title_word_count'].values.reshape(-1,1))

print("After vectorizations")
print(title_word_count_train.shape, y_train.shape)
print(title_word_count_cv.shape, y_cv.shape)
print(title_word_count_test.shape, y_test.shape)
print("="*100)
```

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

Essay word count

In [89]:

```
normalizer = Normalizer()

normalizer.fit(X_train['essay_word_count'].values.reshape(-1,1))

essay_word_count_train = normalizer.transform(X_train['essay_word_count'].values.reshape(-1,1))
essay_word_count_cv = normalizer.transform(X_cv['essay_word_count'].values.reshape(-1,1))
essay_word_count_test = normalizer.transform(X_test['essay_word_count'].values.reshape(-1,1))

print("After vectorizations")
print(essay_word_count_train.shape, y_train.shape)
print(essay_word_count_cv.shape, y_cv.shape)
print(essay_word_count_test.shape, y_test.shape)
print("="*100)
```

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

Assignment 3: Apply KNN

1. **[Task-1] Apply KNN(brute force version) on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)
   - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)
   - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)
2. **Hyper paramter tuning to find best K**
   - Find the best hyper parameter which results in the maximum AUC value
   - Find the best hyper paramter using k-fold cross validation (or) simple cross validation data
   - Use gridsearch-cv or randomsearch-cv or write your own for loops to do this task
3. **Representation of results**
   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure
   - Once you find the best hyper parameter, you need to train your model-M using the best hyper-param. Now, find the AUC on test data and plot the ROC curve on both train and test using model-M.
   - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points
4. **[Task-2]**
   - Select top 2000 features from feature Set 2 using `SelectKBest` and then apply KNN on top of these features
     - ```
       from sklearn.datasets import load_digits
       from sklearn.feature_selection import SelectKBest, chi2
       X, y = load_digits(return_X_y=True)
       X.shape
       X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
       X_new.shape
       ========
       output:
       (1797, 64)
       (1797, 20)
       ```
       ```

5. **Conclusion**

   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

---

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link.

1.5.4 Merging all the above features

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

2. K Nearest Neighbor

2.4 Appling KNN on different kind of featurization as mentioned in the instructions

Apply KNN on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instructions

2.4.1 Applying KNN brute force on BOW, SET 1

In [0]:

```
# Please write all the code with proper documentation
```

In [244]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
#https://github.com/harrismohammed/DonorsChoose.org-knn/blob/master/DonorsChoose_KNN.ipynb
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
title_bow_train,
               text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test, quantity_test,
               prev_projects_test, title_word_count_test, essay_word_count_test, title_bow_test, te
xt_bow_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv,
quantity_cv,
               prev_projects_cv, title_word_count_cv, essay_word_count_cv, title_bow_cv,
text_bow_cv)).tocsr()
```

In [245]:

```
print("Final Data matrix")
```

```
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 10088) (22445,)
(11055, 10088) (11055,)
(16500, 10088) (16500,)
====================================================================================================
```

In [97]:

```
#Finding the best hyper parameter for finding maximum AUC value

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

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

    return y_data_pred
```

In [249]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
a = []
b = []

K= [1, 5, 10, 15, 21, 31, 41, 51, 65, 71]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tr, y_train)

    y_train_pred = batch_predict(neigh, X_tr)
    y_cv_pred = batch_predict(neigh, X_cr)

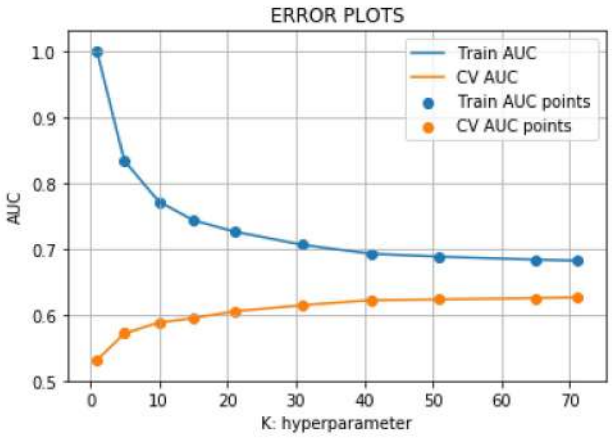

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
    a.append(y_train_pred)
    b.append(y_cv_pred)

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')
```

```
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

```
100%|| 10/10 [19:12<00:00, 115.39s/it]
```



Grid search

In [250]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()

parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv= 5, scoring='roc_auc')

clf.fit(X_tr, y_train)

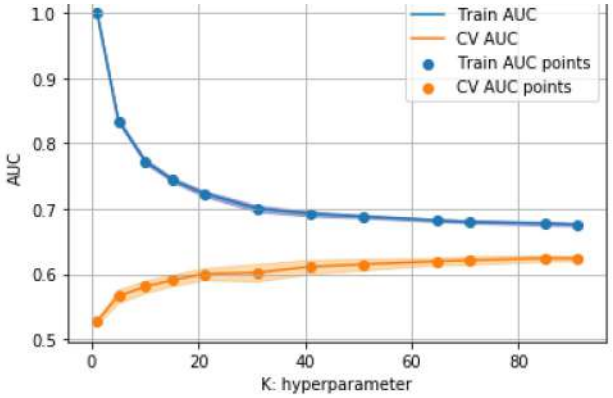
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.3,color='darkblue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,
color='darkorange')

plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [251]:

```
best_k_1 = 91
```

In [252]:

```
#Here we train the model using the best hyper-parameter value

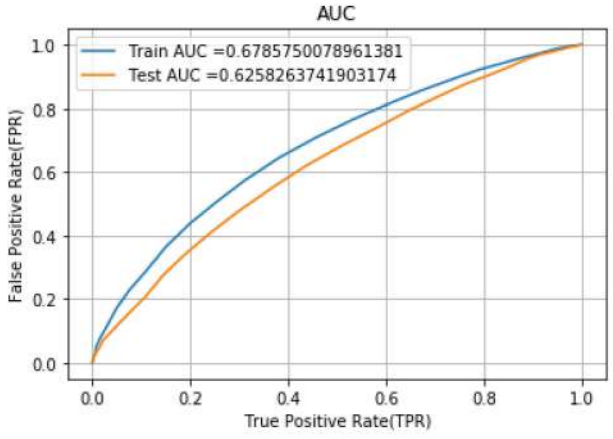

# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k_1)
neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

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

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



In [98]:

```
#Confusion matrix
```

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

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

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

Train data

In [254]:

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

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24905421105704803 for threshold 0.78
[[ 1625  1838]
 [ 4481 14501]]
```

In [255]:

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
train_fpr, train_fpr)),
                                  range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24905421105704803 for threshold 0.78
```

In [256]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[256]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1abc9a70908>
```

Test data

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24987905184047926 for threshold 0.78
[[ 1059  1487]
 [ 3601 10353]]
```

In [258]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, test_f
pr, test_fpr)), range(2),
                                 range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24987905184047926 for threshold 0.78
```

In [259]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[259]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1aba745f550>
```

2.4.2 Applying KNN brute force on TFIDF, SET 2

In [0]:

```
# Please write all the code with proper documentation
```

In [260]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
text_tfidf_train,
               title_tfidf_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test, quantity_test,
               prev_projects_test, title_word_count_test, essay_word_count_test, text_tfidf_test, t
```

```
school_state_categories_one_hot_cv,
                project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv,
quantity_cv,
                prev_projects_cv, title_word_count_cv, essay_word_count_cv, text_tfidf_cv,
title_tfidf_cv)).tocsr()
```

In [261]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 10088) (22445,)
(11055, 10088) (11055,)
(16500, 10088) (16500,)
====================================================================================================
```

In [262]:

```
#Finding the best hyper parameter for finding maximum AUC value
train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tr, y_train)

    y_train_pred = batch_predict(neigh, X_tr)
    y_cv_pred = batch_predict(neigh, X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

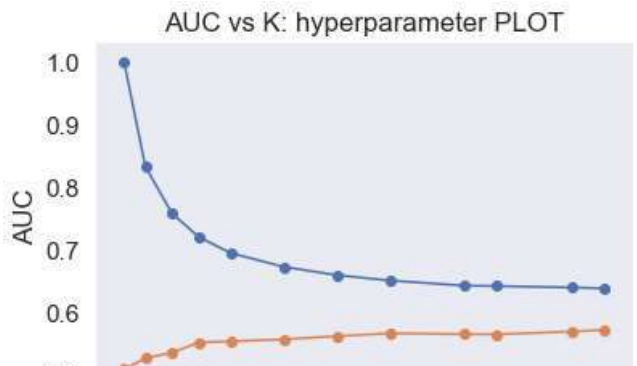
```
100%|██████████| 12/12
[24:38<00:00, 126.84s/it]
```

In [264]:

```
plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs K: hyperparameter PLOT")
plt.grid()
plt.show()
```



K: hyperparameter

Grid search

In [265]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()

parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv= 5, scoring='roc_auc')

clf.fit(X_tr, y_train)

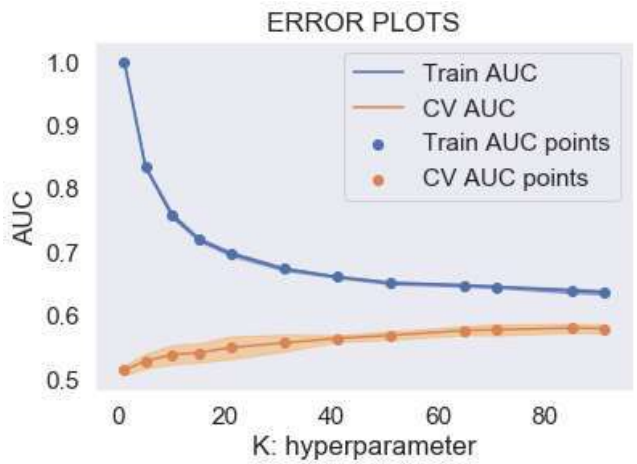
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.3,color='darkblue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,
color='darkorange')

plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [266]:

```
best_k_2 = 85
```

In [267]:

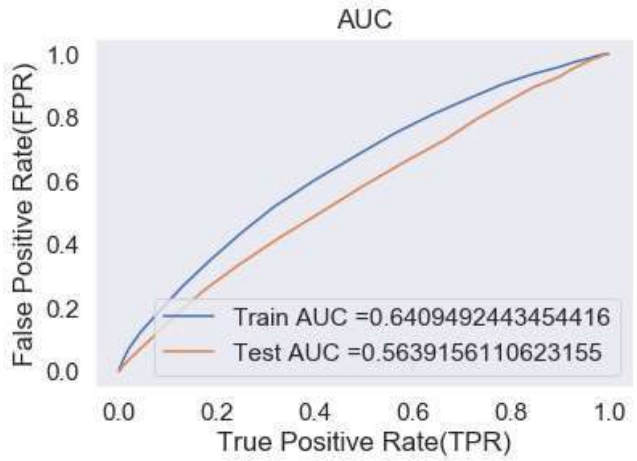
```
#Here we train the model using best hyper-parameter value
```

```
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

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

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



Train data

In [268]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2497338098919404 for threshold 0.835
[[ 1788  1675]
 [ 6105 12877]]
```

In [269]:

```
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
train_fpr, train_fpr)),
                                    range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.2497338098919404 for threshold 0.835
```

In [270]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[270]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1abc8331cc0>
```

Test data

In [271]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999861156449527 for threshold 0.859
[[1500 1046]
 [6990 6964]]
```

In [272]:

```
conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, tes
t_fpr, test_fpr)),
                                   range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999861156449527 for threshold 0.859
```

In [273]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[273]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ab9a2eaa20>
```

2.4.3 Applying KNN brute force on AVG W2V, SET 3

```
# Please write all the code with proper documentation
```

In [276]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
avg_w2v_vectors_train,
               avg_w2v_vectors_titles_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test,
               quantity_test, prev_projects_test, title_word_count_test, essay_word_count_test, avg
_w2v_vectors_test,
               avg_w2v_vectors_titles_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv,

               quantity_cv,prev_projects_cv, title_word_count_cv, essay_word_count_cv,
avg_w2v_vectors_cv,
               avg_w2v_vectors_titles_cv)).tocsr()
```

In [277]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 705) (22445,)
(11055, 705) (11055,)
(16500, 705) (16500,)
====================================================================================================
```

In [278]:

```
#Find the best hyper parameter which results in the maximum AUC value

train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tr, y_train)

    y_train_pred = batch_predict(neigh, X_tr)
    y_cv_pred = batch_predict(neigh, X_cr)

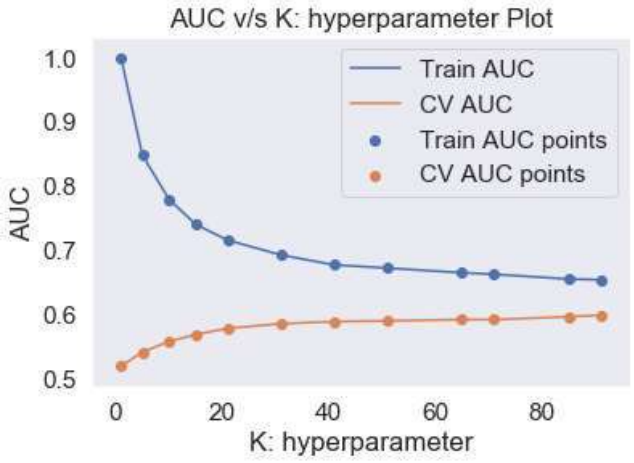
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

```
100%|██████████| 12/12
[5:02:17<00:00, 1596.50s/it]
```

In [279]:

```
plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC v/s K: hyperparameter Plot")
plt.grid()
plt.show()
```



Grid search

In [ ]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = KNeighborsClassifier()

parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv= 5, scoring='roc_auc')

clf.fit(X_tr, y_train)

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

plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.3,color='darkblue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,
color='darkorange')

plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

In [281]:

```
best_k_3 = 91
```

In [282]:

```
#Training the model using the best hyper-parameter value



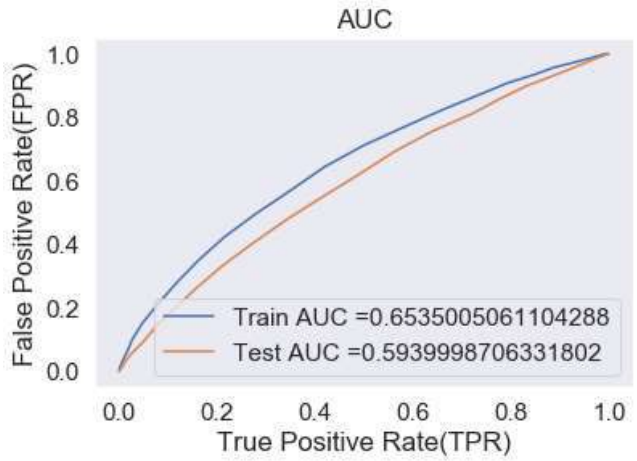
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve

neigh = KNeighborsClassifier(n_neighbors=best_k_3)
neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

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

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



Train data

In [283]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.835
[[ 1732  1731]
 [ 5477 13505]]
```

In [284]:

```
),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.835
```

In [285]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[285]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ab9b6f9e10>
```

Test data

In [286]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2499605067234218 for threshold 0.846
[[1289 1257]
 [5251 8703]]
```

In [287]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds,
                                                                    test_fpr, test_fpr)),
range(2),range(2))
```

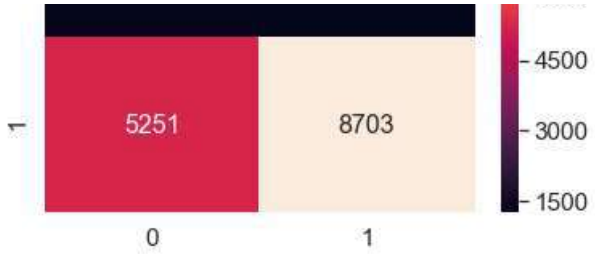
```
the maximum value of tpr*(1-fpr) 0.2499605067234218 for threshold 0.846
```

In [288]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[288]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac39b09668>
```



2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [289]:

```
# Please write all the code with proper documentation
```

In [92]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
tfidf_w2v_vectors_train,
               tfidf_w2v_vectors_titles_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
_test,
               quantity_test, prev_projects_test, title_word_count_test, essay_word_count_test, tfi
df_w2v_vectors_test,
               tfidf_w2v_vectors_titles_test)).tocsr()

X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv,

               quantity_cv, prev_projects_cv, title_word_count_cv, essay_word_count_cv,
tfidf_w2v_vectors_cv,
               tfidf_w2v_vectors_titles_cv)).tocsr()
```

In [93]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 705) (22445,)
(11055, 705) (11055,)
(16500, 705) (16500,)
====================================================================================================
```

In [97]:

```
#Finding best hyper parameter which results in the maximum AUC value


import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
train_auc = []
```

```
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tr, y_train)

    y_train_pred = batch_predict(neigh, X_tr)
    y_cv_pred = batch_predict(neigh, X_cr)

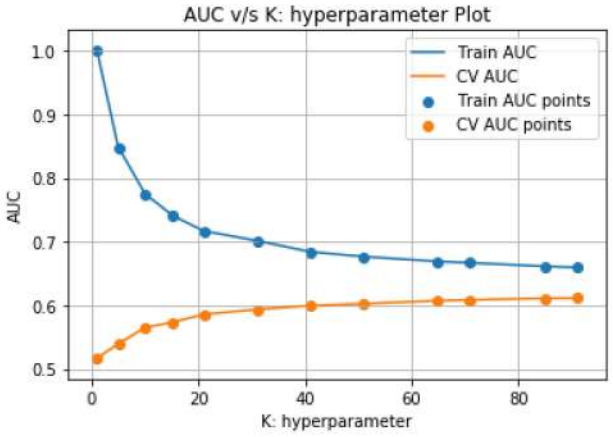
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC v/s K: hyperparameter Plot")
plt.grid()
plt.show()
```

```
100%|██████████| 12/12 [7:01:56<00:00, 2686.38s/it]
```



Grid search

In [265]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

neigh = KNeighborsClassifier()
parameters = {'n_neighbors':[1, 5, 10, 15, 21, 33, 41, 51, 65, 71, 85, 91]}
clf = GridSearchCV(neigh, parameters, cv=5, scoring='roc_auc')
clf.fit(X_tr, y_train)

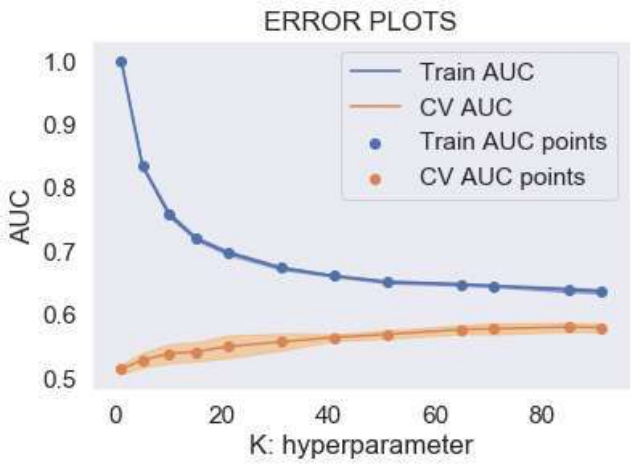
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
```

```
plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC v/s K: hyperparameter Plot - using GridSearchcv")
plt.grid()
plt.show()
```



For this grid search as it taking so much i left it without processing it following advice given by assignments department when i contacted them.so please consider it.

In [98]:

```
best_k_4 = 85
```

In [99]:

```
#Here we train model using the best hyper-parameter value

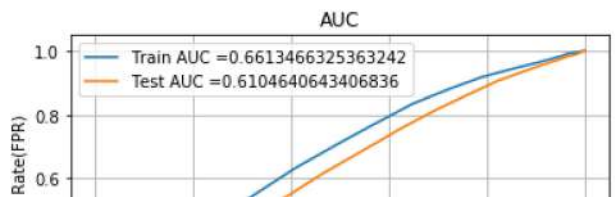
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve

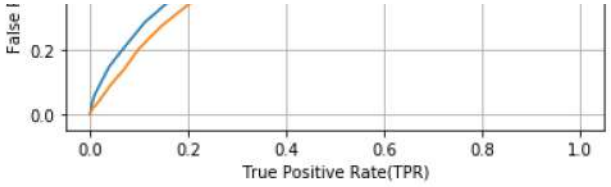
neigh = KNeighborsClassifier(n_neighbors=best_k_4)
neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

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

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





Train data

In [102]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999247438100008 for threshold 0.835
[[ 1741  1722]
 [ 5517 13465]]
```

In [103]:

```
conf_matr_df_train_3 = pd.DataFrame(confusion_matrix(y_train,
                                                     predict(y_train_pred, tr_thresholds, train_fpr,
train_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999247438100008 for threshold 0.835
```

In [104]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_3, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[104]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bffbdfc0f0>
```

Test data

In [105]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2485183850458708 for threshold 0.847
[[1371 1175]
 [5418 8536]]
```

In [106]:

```
conf_matr_df_test_3 = pd.DataFrame(confusion_matrix(y_test,
                                    predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), range
2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.2485183850458708 for threshold 0.847
```

In [107]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_3, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[107]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bffc693ba8>
```

2.5 Feature selection with `SelectKBest`

In [0]:

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

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

In [90]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train,
               project_grade_categories_one_hot_train, teacher_prefix_categories_one_hot_train, pri
ce_train,
               quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train,
text_tfidf_train,
               title_tfidf_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test,
               project_grade_categories_one_hot_test, teacher_prefix_categories_one_hot_test, price
```

```
itle_tfidf_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv,
               project_grade_categories_one_hot_cv, teacher_prefix_categories_one_hot_cv, price_cv,
quantity_cv,
               prev_projects_cv, title_word_count_cv, essay_word_count_cv, text_tfidf_cv,
title_tfidf_cv)).tocsr()
```

In [94]:

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
selectkbest = SelectKBest(chi2, k=20)
X_tr_new = selectkbest.fit_transform(X_tr, y_train)
X_te_new = selectkbest.transform(X_te)
X_cr_new = selectkbest.transform(X_cr)
```

In [95]:

```
print("Final Data matrix")
print(X_tr_new.shape)
print(X_cr_new.shape)
print(X_te_new.shape)
print("="*100)
```

```
Final Data matrix
(22445, 20)
(11055, 20)
(16500, 20)
====================================================================================================
```

In [99]:

```
#Find the best hyper parameter which results in the maximum AUC value

import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score


train_auc = []
cv_auc = []

K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tr_new, y_train)

    y_train_pred = batch_predict(neigh, X_tr_new)
    y_cv_pred = batch_predict(neigh, X_cr_new)

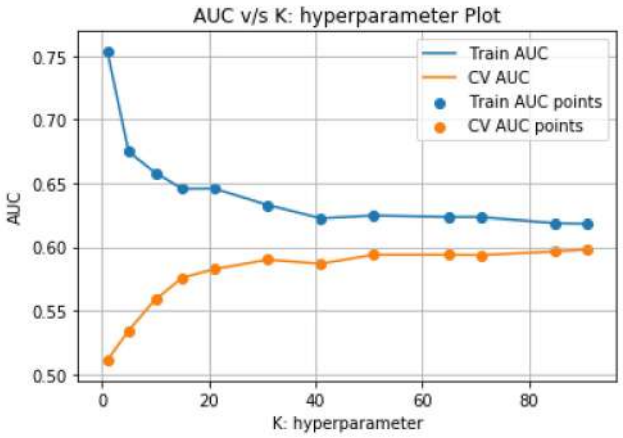
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(K, train_auc, label='Train AUC')
plt.plot(K, cv_auc, label='CV AUC')

plt.scatter(K, train_auc, label='Train AUC points')
plt.scatter(K, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC v/s K: hyperparameter Plot")
plt.grid()
plt.show()
```

[11:04<00:00, 56.90s/it]



Grid search

In [265]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

neigh = KNeighborsClassifier()

parameters = {'n_neighbors':[1, 5, 10, 15, 21, 33, 41, 51, 65, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv=5, scoring='roc_auc')
clf.fit(X_tr_new, y_train)

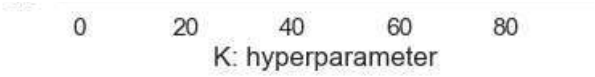
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,
color='darkorange')

plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC v/s K: hyperparameter Plot - GridsearchCV")
plt.grid()
plt.show()
```



For this grid search as it taking so much i left it without processing it following advice given by assignments department when i contacted them.so please consider it.

In [113]:

```
best_k_5 = 85
```

In [114]:

```
#Train model using the best hyper-parameter value

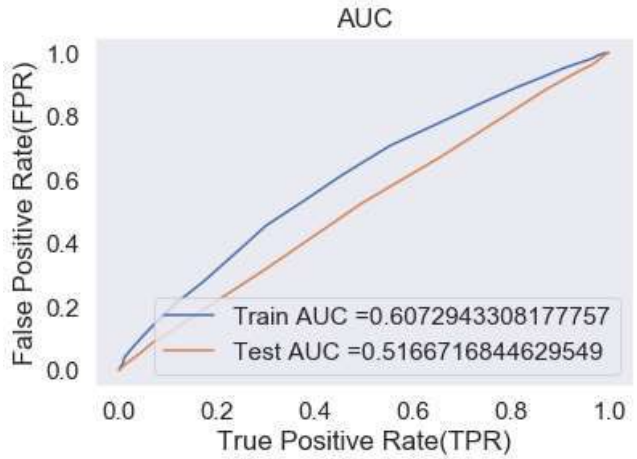

# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve

neigh = KNeighborsClassifier(n_neighbors=best_k_5)
neigh.fit(X_tr_new, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr_new)
y_test_pred = batch_predict(neigh, X_te_new)

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

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



Train data

In [115]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================

Train confusion matrix
```

In [116]:

```
conf_matr_df_train_4 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred,
                                                    tr_thresholds, train_fpr, train_fpr)), range(2),range(2
)
```

```
the maximum value of tpr*(1-fpr) 0.2480095467375962 for threshold 0.835
```

In [117]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_4, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[117]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2bffbdeae10>
```

Test data

In [118]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24998133325599237 for threshold 0.847
[[ 2378   168]
 [12779  1175]]
```

In [119]:

```
conf_matr_df_test_4 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred,
                                                    tr_thresholds, test_fpr, test_fpr)), ra
e(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24998133325599237 for threshold 0.847
```

In [120]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_4, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[120]:

3. Conclusions

In [0]:

```
# Please compare all your models using Prettytable library
```

In [121]:

```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable

a = PrettyTable()
a.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]

a.add_row(["BOW", "Brute", 91, 0.63])
a.add_row(["TFIDF", "Brute", 85, 0.57])
a.add_row(["AVG W2V", "Brute", 91, 0.6])
a.add_row(["TFIDF W2V", "Brute", 85, 0.55])
a.add_row(["TFIDF", "Top 2000", 85, 0.51])

print(x)
```

```
+------------+----------+-----------------+------+
| Vectorizer |  Model   | Hyper Parameter | AUC  |
+------------+----------+-----------------+------+
|    BOW     |  Brute   |        91       | 0.63 |
|   TFIDF    |  Brute   |        85       | 0.57 |
|  AVG W2V   |  Brute   |        91       | 0.6  |
| TFIDF W2V  |  Brute   |        85       | 0.55 |
|   TFIDF    | Top 2000 |        85       | 0.51 |
+------------+----------+-----------------+------+
```