

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		Desc
project_id		A unique identifier for the proposed project. Example: p0
		Title of the project. Example:
project_title	• •	Art Will Make You H First Grad
		Grade level of students for which the project is targeted. One of the following enumerated values
project_grade_category	• • • •	Grades P
		Grade
		Grade
		Grades
project_subject_categories	• • • • • • • • • •	One or more (comma-separated) subject categories for the project from the following enumerated list of values
		Applied Learning
		Care & Health
		Health & Safety
		History & Culture
		Literacy & Language
		Math & Science
		Music & The Arts
		Special Education
		World Languages
	• •	Example:
		Music & The Arts, Literacy & Language, Math & Science

Feature	Desc
<code>school_state</code>	State where school is located (Two-letter U.S. postal codes) (https://en.wikipedia.org/wiki/List of U.S. state abbreviations#Postal codes) Example: CA
<code>project_subject_subcategories</code>	One or more (comma-separated) subject subcategories for the project. Example: Lit, Literature & Writing, Social Sci
<code>project_resource_summary</code>	An explanation of the resources needed for the project. Example: My students need hands on literacy materials to meet sensory needs!<
<code>project_essay_1</code>	First application
<code>project_essay_2</code>	Second application
<code>project_essay_3</code>	Third application
<code>project_essay_4</code>	Fourth application
<code>project_submitted_datetime</code>	Datetime when project application was submitted. Example: 2016-01-12:43:5
<code>teacher_id</code>	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4ff1c
<code>teacher_prefix</code>	Teacher's title. One of the following enumerated values: • • • • • • Tea
<code>teacher_number_of_previously_posted_projects</code>	Number of project applications previously submitted by the same teacher. Example:

* 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
<code>id</code>	A <code>project_id</code> value from the <code>train.csv</code> file. Example: p036502
<code>description</code>	Description of the resource. Example: Tenor Saxophone Reeds, Box of 25
<code>quantity</code>	Quantity of the resource required. Example: 3
<code>price</code>	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
<code>project_is_approved</code>	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 string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

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

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_limit = []
project_limit = project_data[:20000]

print("Number of 20k data points in train data", project_limit.shape)
```

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

The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'

'project_submitted_datetime' 'project_grade_category'
'project_subject_categories' 'project_subject_subcategories'
'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
'project_essay_4' 'project_resource_summary'
'teacher_number_of_previously_posted_projects' 'project_is_approved']

Number of 20k data points in train data (20000, 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_limit.columns)]

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

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

project_limit.head(2)
```

Out[4]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2019-08-00:50:00
7176	79341	p091436	bb2599c4a114d211b3381abe9f899bf8	Mrs.	OH	2019-07:20:00

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_limit)):
    a = project_limit["project_grade_category"][i].replace(" ", "_")
    project_grade_category.append(a)
```

In [7]:

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

In [8]:

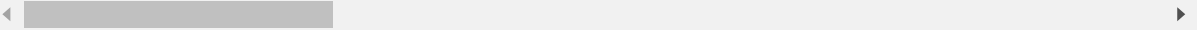
```
project_limit["project_grade_category"] = project_grade_category
```

In [9]:

```
project_limit.head(5)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	l
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2 0. 00:5
7176	79341	p091436	bb2599c4a114d211b3381abe9f899bf8	Mrs.	OH	2 0. 07:2
5145	50256	p203475	63e9a9f2c9811a247f1aa32ee6f92644	Mrs.	CA	2 0. 08:4
2521	164738	p248458	40da977f63fb3d85589a063471304b11	Ms.	NJ	2 0. 09:3
5364	14044	p002546	91dacb4ab5754671f342b4a12abf3cfb	Mr.	CO	2 0. 10:1



In [10]:

```

project_grade_category = []

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

project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data["project_grade_category"] = project_grade_category
project_data.head(5)

```

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

1.2 preprocessing of project_subject_categories

In [11]:

```

categories = list(project_limit['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019
# 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 categories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the category based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ', '') # we are placing all the ' '(space) with ''(empty) ex:"Math
        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_limit['clean_categories'] = cat_list
project_limit.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_limit['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_limit['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

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

project_limit['clean_subcategories'] = sub_cat_list
project_limit.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_limit['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]))

```

Feature "Number of Words in Title"

In [13]:

```

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

project_limit["title_word_count"] = title_word_count
project_limit.head(5)

```

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	I
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2 0. 00:5
7176	79341	p091436	bb2599c4a114d211b3381abe9f899bf8	Mrs.	OH	2 0. 07:2
5145	50256	p203475	63e9a9f2c9811a247f1aa32ee6f92644	Mrs.	CA	2 0. 08:4
2521	164738	p248458	40da977f63fb3d85589a063471304b11	Ms.	NJ	2 0. 09:3
5364	14044	p002546	91dacb4ab5754671f342b4a12abf3cfb	Mr.	CO	2 0. 10:1

1.3 Text preprocessing

In [14]:

```

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

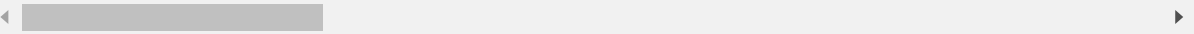
```

In [15]:

```
project_limit.head(2)
```

Out[15]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	I
473	100660 p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	21:00:5
7176	79341 p091436	bb2599c4a114d211b3381abe9f899bf8	Mrs.	OH	21:07:2



Number of Words in Essay

In [16]:

```

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

project_limit["essay_word_count"] = essay_word_count

project_limit.head(5)

```

Out[16]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	I
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2 0. 00:5
7176	79341	p091436	bb2599c4a114d211b3381abe9f899bf8	Mrs.	OH	2 0. 07:2
5145	50256	p203475	63e9a9f2c9811a247f1aa32ee6f92644	Mrs.	CA	2 0. 08:4
2521	164738	p248458	40da977f63fb3d85589a063471304b11	Ms.	NJ	2 0. 09:3
5364	14044	p002546	91dacb4ab5754671f342b4a12abf3cfb	Mr.	CO	2 0. 10:1

1.7 Test - Train Split

In [17]:

```

# train test split

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_limit, project_limit['project_i
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=

```

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]:

```
# printing some random reviews
print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[150])
print("="*50)
print(X_train['essay'].values[1000])
print("="*50)
print(X_train['essay'].values[5000])
print("="*50)
```

My students demographics comprises of severe socio-economically disadvantage d. 100 percent receive free breakfast and lunch. Our population is divided into two groups. The Emotionally Disturbed and the Autistic. My kids love sports and both populations are capable of performing on a competitive platform. They are eager to learn new sports. My students needs to learn sportsmanship through behavior. My students need to learn through diverse means and modalities such as competitive sports and organized play. Structure is derived and a social platform is structured through sports.\r\n\r\nWe are attempting to start team sports with our alternate assessment students through modifications that enable them to actively, and fully be engaged in competition. These materials are essential for our school to expeditiously implement teams sports in an unique way. This equipment will allow us the flexibility to encompass all of our student athletic abilities and assist and developing and enriching their social skills. This will not only enrich the lives of the students but that of their parents and ultimately strengthen the communities that the students reside in. Our school is designed to provide the extra opportunities for social and academic development.nannan

=====

Do you remember the feeling before a big exam when you just knew you were going to ace it? Days, weeks, or maybe months of practice made you stronger and sharper. Last year, my students worked daily to become stronger readers, sharper writers, and original thinkers. They were ready to make our school, their families, and themselves proud.\r\n\r\nYet, when the time came to show that they had learned, they fell short; not because they weren't strong, sharp, or original but they hadn't had the opportunity to show their learning on a technology-based test. \r\n\r\n It is time to make this change and provide them with the support they need to achieve! My students were unprepared simply because they hadn't had enough exposure to technology to learn the basic typing skills needed to communicate on this state exam. They are wonderful, intelligent scholars, who arrive to school curious and ready to work. This year, they need more opportunities to use technology to learn basic typing skills so they can demonstrate their amazing knowledge! \r\n\r\n\r\nMy students have not had the opportunity to learn basic typing skills. This is essential in today's classroom! Students are expected to demonstrate a year's worth of learning on a computer-based test that requires typed essays. They are also limited in the time they have to accomplish this goal! The sad reality became that my creative thoughtful writers were not able to finish the exam, because they did not know how to type. As they looked for the tab key to properly indent, quotation marks to properly cite evidence, and even the "\r\n" key to start a new paragraph, the time ticked by. \r\n\r\nThis year, I want my students to feel prepared with content and with the test format; this requires a knowledge of typing! With these cases, I can utilize the Kindle Fire tablets that have been previously gifted to my classroom for a second purpose. In addition to being a great resource for reading differentiation, if we obtain these cases with keyboards, my students will have the opportunity to learn how to type their responses to reading. This an opportunity they need and deserve to feel better prepared for state testing.nannan

=====

My students are funny, curious and very expressive! They say what they feel, but they always help each other! The school that they attend is a Title I school! The majority of my students eat free meals at school! Many of them at a young age have emigrated to the United States with their parents for a better life. Their life is hard not knowing how to speak English and being in a new country trying to find their way. Our school is focused on giving all the students the best leg up they can get due to the community that they are growing up in. STEM kits early in their academic careers can help them build basic knowledge before the next year where they become more in depth. But to keep it on level and make meaningful learning these kits come based on fairy tales. Fairy tales are always interesting to my small students, and to mix in STEM will allow them to grow in more than one ways. I would be so excited to be able to send them to the next level with this content knowledge!nannan

=====

Our school represents a large variety of students from various countries all over the world. One hundred percent of the students at my school receive free or reduced-price lunch and we are a schoolwide recipient of Title I funds.

\r\n\r\nOne of my favorite parts of my job? I love seeing the change that my students undergo over the course of the year. I am fortunate enough to see kids develop ways to express kindness each and every day. They use manners that weren't present in August. I have seen students who loathe reading turn into book lovers. A great part... hearing students say that math is their favorite subject... because that was not true for me as a child. My classroom is full of faces and personalities that will impact our future world. As a teacher, I believe it is my job to do everything I can to make sure that impact will be a positive impact. These supplies are all basic, necessary supplies for an elementary school student. Students are given what they need to be successful at school even if they cannot bring it from home. If students want to "upgrade" to fancy pencils, they want their own pencils that won't be for the "community", funky erasers, etc. they may use their behavior points or class money to do so... a lot like real life! You may be given what you MUST have, but you always have the chance to work a little harder and have your own things that may be a bit nicer than what is given out. This year in particular, I have loved seeing students use the resources they have earned to better themselves... even if it is in pencils and erasers. Developing responsibility, realizing there are costs for wants and needs that must be balanced, and not always knowing when an expense will come up (say, if a neighbor has used all your tables pencils in one day!) are important life lessons that I teach by having students use their resources to "buy" things they need and want.nannan

=====

In [20]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [21]:

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

My students demographics comprises of severe socio-economically disadvantage d. 100 percent receive free breakfast and lunch. Our population is divided into two groups. The Emotionally Disturbed and the Autistic. My kids love sports and both populations are capable of performing on a competitive platform. They are eager to learn new sports. My students needs to learn sportsmanship through behavior. My students need to learn through diverse means and modalities such as competitive sports and organized play. Structure is derived and a social platform is structured through sports.\r\n\r\nWe are attempting to start team sports with our alternate assessment students through modifications that enable them to actively, and fully be engaged in competition. These materials are essential for our school to expeditiously implement teams sports in a unique way. This equipment will allow us the flexibility to encompass all of our student athletic abilities and assist and developing and enriching their social skills. This will not only enrich the lives of the students but that of their parents and ultimately strengthen the communities that the students reside in. Our school is designed to provide the extra opportunities for social and academic development.nannan

=====

In [22]:

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

My students demographics comprises of severe socio-economically disadvantaged. 100 percent receive free breakfast and lunch. Our population is divided into two groups. The Emotionally Disturbed and the Autistic. My kids love sports and both populations are capable of performing on a competitive platform. They are eager to learn new sports. My students needs to learn sportsmanship through behavior. My students need to learn through diverse means and modalities such as competitive sports and organized play. Structure is derived and a social platform is structured through sports. We are attempting to start team sports with our alternate assessment students through modifications that enable them to actively, and fully be engaged in competition. These materials are essential for our school to expeditiously implement teams sports in an unique way. This equipment will allow us the flexibility to encompass all of our student athletic abilities and assist and developing and enriching their social skills. This will not only enrich the lives of the students but that of their parents and ultimately strengthen the communities that the students reside in. Our school is designed to provide the extra opportunities for social and academic development.nannan

In [23]:

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

My students demographics comprises of severe socio economically disadvantaged d 100 percent receive free breakfast and lunch Our population is divided into two groups The Emotionally Disturbed and the Autistic My kids love sports and both populations are capable of performing on a competitive platform The y are eager to learn new sports My students needs to learn sportsmanship through behavior My students need to learn through diverse means and modalities such as competitive sports and organized play Structure is derived and a social platform is structured through sports We are attempting to start team sports with our alternate assessment students through modifications that enable e them to actively and fully be engaged in competition These materials are essential for our school to expeditiously implement teams sports in an unique way This equipment will allow us the flexibility to encompass all of our student athletic abilities and assist and developing and enriching their social skills This will not only enrich the lives of the students but that of their parents and ultimately strengthen the communities that the students reside in n Our school is designed to provide the extra opportunities for social and academic development nannan

In [24]:

```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'each', 'both', 'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn't', 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'won', "won't", 'wouldn', "wouldn't"]
```

In [25]:

```
# Combining all the above students
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%|██████████| 8978/8978 [00:05<00:00, 1513.59it/s]

In [26]:

```
# after preprocessing
preprocessed_essays_train[0]
```

Out[26]:

'students demographics comprises severe socio economically disadvantaged 100 percent receive free breakfast lunch population divided two groups emotional ly disturbed autistic kids love sports populations capable performing competitive platform eager learn new sports students needs learn sportsmanship behavior students need learn diverse means modalities competitive sports organized play structure derived social platform structured sports attempting start team sports alternate assessment students modifications enable actively fully engaged competition materials essential school expeditiously implement teams sports unique way equipment allow us flexibility encompass student athletic abilities assist developing enriching social skills not enrich lives students parents ultimately strengthen communities students reside school designed provide extra opportunities social academic development nannan'

Preprocessed Test data (Text)

In [27]:

```

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%|██████████| 6600/6600 [00:04<00:00, 1542.38it/s]

In [28]:

```
preprocessed_essays_test[0]
```

Out[28]:

'utmost pleasure work outstanding group third graders year students come various backgrounds cultures many students receive aid school free reduced lunch majority students bilingual despite differences tight knit community loves supports one another academically emotionally given amazing opportunity loop students upcoming school year excited accompanying students fourth grade 16 17 school year looping class shown great academic success allows students teachers form meaningful bond unfortunately also takes lot work adapt classroom new grade place everything everything place benjamin franklin many students come large families chaotic situations school quiet safe place express peacefully strive keep classroom neat orderly provide students stability need disorganized room sign chaotic mind donation room neat orderly full students ready learn nannan'

Preprocessed Cross Validation data (Text)

In [29]:

```

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%|██████████| 4422/4422 [00:02<00:00, 1533.60it/s]

In [30]:

```
preprocessed_essays_cv[0]
```

Out[30]:

```
'technology best brings people together matt mullenweg classroom filled 28 f
un fabulous fifth graders students active love move around love use technolo
gy desire fully integrate technology aspects classroom better reach students
live rural county school intermediate school county school serves fifth sixt
h graders district 800 students currently classroom two chromebooks belong u
s goal able provide student opportunity work technology throughout day addit
ion chromebooks able take learning next level students would able small grou
ps individualize learning bring learning life google drawings 2 touchscreen
chromebooks help creative students creatively gifted students show love lear
ning new way district also implements google apps education within schools a
dding touch enabled chromebooks students finally able use google drawings br
ing learning next level kids best deserve best every aspect life especially
education help help students achieve best partnering not help class also man
y students come thank advance generous support partnering fabulous fifth gra
ders nannan'
```

1.4 Preprocessing of `project_title`

Preprocessing of Project Title for Train data

In [31]:

```
# similarly you can preprocess the titles also
print(project_limit['project_title'].values[0])
print("="*50)
print(project_limit['project_title'].values[150])
print("="*50)
print(project_limit['project_title'].values[1000])
print("="*50)
print(project_limit['project_title'].values[5000])
print("="*50)
```

```
Flexible Seating for Flexible Learning
```

```
=====
```

```
Color Printer Ink for Next Year
```

```
=====
```

```
Cooperative Learning in 4th Grade!
```

```
=====
```

```
Funtastic Phonics
```

```
=====
```

In [32]:

```
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%|██████████| 8978/8978 [00:00<00:00, 33231.17it/s]

In [33]:

```
preprocessed_titles_train[0]
```

Out[33]:

'sports matter'

Preprocessing of Project Title for Test data

In [34]:

```
preprocessed_titles_test = []

for titles in tqdm(X_test["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_test.append(title.lower().strip())
```

100%|██████████| 6600/6600 [00:00<00:00, 33216.07it/s]

In [35]:

```
preprocessed_titles_test[0]
```

Out[35]:

'a place everything everything it place'

Preprocessing of Project Title for Train data

In [36]:

```

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%|██████████| 4422/4422 [00:00<00:00, 31547.76it/s]

In [37]:

```
preprocessed_titles_cv[0]
```

Out[37]:

'technology at the tip of our fingers'

1.5 Preparing data for models

In [38]:

```
project_limit.columns
```

Out[38]:

```

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
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)
- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

1.5.1 Vectorizing Categorical data

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

One Hot Encoding For Categories

In [39]:

```
# 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, binarize=False)
categories_one_hot = vectorizer.fit_transform(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 after one hot encoding train ", categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding test ", categories_one_hot_test.shape)
print("Shape of matrix after one hot encoding CV ", categories_one_hot_cv.shape)

['Warmth', 'Math_Science', 'Music_Arts', 'History_Civics', 'Health_Sports',
 'Care_Hunger', 'SpecialNeeds', 'Literacy_Language', 'AppliedLearning']
Shape of matrix after one hot encoding train (8978, 9)
Shape of matrix after one hot encoding test (6600, 9)
Shape of matrix after one hot encoding CV (4422, 9)
```

One Hot Encoding For Sub-Categories

In [40]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
sub_categories_one_hot = vectorizer.fit_transform(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 after one hot encoding Train",sub_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding Test",sub_categories_one_hot_test.shape)
print("Shape of matrix after one hot encoding CV",sub_categories_one_hot_cv.shape)
```

```
['PerformingArts', 'Literature_Writing', 'SocialSciences', 'Mathematics', 'E
SL', 'College_CareerPrep', 'Warmth', 'Economics', 'EarlyDevelopment', 'Extra
curricular', 'SpecialNeeds', 'Health_Wellness', 'ForeignLanguages', 'Music',
'AppliedSciences', 'CommunityService', 'Other', 'Literacy', 'ParentInvolveme
nt', 'FinancialLiteracy', 'Gym_Fitness', 'History_Geography', 'NutritionEduc
ation', 'VisualArts', 'EnvironmentalScience', 'Care_Hunger', 'Civics_Governm
ent', 'CharacterEducation', 'Health_LifeScience', 'TeamSports']
```

```
Shape of matrix after one hot encoding Train (8978, 30)
```

```
Shape of matrix after one hot encoding Test (6600, 30)
```

```
Shape of matrix after one hot encoding CV (4422, 30)
```

One Hot Encoding For State

In [41]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [42]:

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

In [43]:

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

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

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("Shape of matrix after one hot encoding Train ", school_state_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding Test", school_state_categories_one_hot_test.shape)
print("Shape of matrix after one hot encoding CV", school_state_categories_one_hot_cv.shape)

['VT', 'IL', 'HI', 'PA', 'NE', 'ID', 'CA', 'KY', 'WI', 'MS', 'FL', 'SC', 'ME', 'WY', 'IN', 'UT', 'NJ', 'AL', 'NY', 'ND', 'WA', 'LA', 'MD', 'DE', 'TN', 'CO', 'OR', 'TX', 'NC', 'NH', 'CT', 'GA', 'AZ', 'NV', 'AK', 'MI', 'OK', 'KS', 'VA', 'IA', 'WV', 'SD', 'MT', 'NM', 'RI', 'MN', 'MA', 'MO', 'OH', 'AR', 'DC']
Shape of matrix after one hot encoding Train (8978, 51)
Shape of matrix after one hot encoding Test (6600, 51)
Shape of matrix after one hot encoding CV (4422, 51)
```

One Hot Encoding For teacher_prefix

In [44]:

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

In [45]:

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

In [46]:

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

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), 1
vectorizer_grade.fit(X_train['project_grade_category'].values)

project_grade_categories_one_hot_train = vectorizer_grade.transform(X_train['project_grade_
project_grade_categories_one_hot_test = vectorizer_grade.transform(X_test['project_grade_ca
project_grade_categories_one_hot_cv = vectorizer_grade.transform(X_cv['project_grade_catego

print(vectorizer_grade.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",project_grade_categories_one
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_one_h
print("Shape of matrix of Cross Validation data after one hot encoding ",project_grade_cate
```

```
['Grades_6-8', 'Grades_3-5', 'Grades_PreK-2', 'Grades_9-12']
Shape of matrix of Train data after one hot encoding (8978, 4)
Shape of matrix of Test data after one hot encoding (6600, 4)
Shape of matrix of Cross Validation data after one hot encoding (4422, 4)
```

One Hot Encoding For project_grade_category

In [47]:

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

In [48]:

```
teacher_prefix_cat_dict = dict(my_counter)
sorted_teacher_prefix_cat_dict = dict(sorted(teacher_prefix_cat_dict.items(), key=lambda kv
```

In [49]:

```

## 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 how to tackle such discrepancies.
## https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-

vectorizer_teacher = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys()))
vectorizer_teacher.fit(X_train['teacher_prefix'].values.astype("U"))

teacher_prefix_categories_one_hot_train = vectorizer_teacher.transform(X_train['teacher_pre
teacher_prefix_categories_one_hot_test = vectorizer_teacher.transform(X_test['teacher_prefi
teacher_prefix_categories_one_hot_cv = vectorizer_teacher.transform(X_cv['teacher_prefix']).

print(vectorizer_teacher.get_feature_names())

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

```

```

['Ms.', 'Teacher', 'Mr.', 'Mrs.', 'Dr.', 'nan']
Shape of matrix after one hot encoding (8978, 6)
Shape of matrix after one hot encoding (6600, 6)
Shape of matrix after one hot encoding (4422, 6)

```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words Essay Train Data

In [50]:

```

# We are considering only the words which appeared in at least 10 documents(rows or project
vectorizer_bow_essay = CountVectorizer(min_df=10)

vectorizer_bow_essay.fit(preprocessed_essays_train)

text_bow_train = vectorizer_bow_essay.transform(preprocessed_essays_train)

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

```

```

Shape of matrix after one hot encoding (8978, 5730)

```

Bag of words Essay Test Data

In [51]:

```

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

```

```

Shape of matrix after one hot encoding (6600, 5730)

```

Bag of words Essay CV Data

In [52]:

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

Shape of matrix after one hot encoding (4422, 5730)

Bag of words Title Train Data

In [53]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_bow_title = CountVectorizer(min_df=10)

vectorizer_bow_title.fit(preprocessed_titles_train)

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

Shape of matrix after one hot encoding (8978, 610)

Bag of words Title Test Data

In [54]:

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

Shape of matrix after one hot encoding (6600, 610)

Bag of words Title CV Data

In [55]:

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

Shape of matrix after one hot encoding (4422, 610)

1.5.2.2 TFIDF vectorizer Essay Train Data

In [56]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

Shape of matrix after one hot encoding (8978, 5730)

TFIDF vectorizer Essay Test Data

In [57]:

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

Shape of matrix after one hot encoding (6600, 5730)

TFIDF vectorizer Essay CV Data

In [58]:

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

Shape of matrix after one hot encoding (4422, 5730)

TFIDF vectorizer Titles Train Data

In [59]:

```
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 (8978, 610)

TFIDF vectorizer Titles Test Data

In [60]:

```
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 (6600, 610)

TFIDF vectorizer Titles CV Data

In [61]:

```
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 (4422, 610)

1.5.2.3 Using Pretrained Models: Avg W2V

In [62]:

Reading glove vectors in python: <https://stackoverflow.com/a/38230349/4084039>

```

def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    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

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

words = []
for i in preprocessed_essays_train :
    words.extend(i.split(' '))

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

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our corpus", \
      len(inter_words), "(" , np.round(len(inter_words)/len(words)*100,3), "%)")

words_corpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_corpus[i] = model[i]
print("word 2 vec length", len(words_corpus))

# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickl

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

```

Loading Glove Model

```

-----
FileNotFoundError                                Traceback (most recent call last)
<ipython-input-62-40504d584192> in <module>
     12     return model
     13
--> 14 model = loadGloveModel('glove.42B.300d.txt')
     15
     16

<ipython-input-62-40504d584192> in loadGloveModel(gloveFile)
      2 def loadGloveModel(gloveFile):
      3     print ("Loading Glove Model")

```

```

----> 4     f = open(gloveFile, 'r', encoding="utf8")
      5     model = {}
      6     for line in tqdm(f):

```

FileNotFoundError: [Errno 2] No such file or directory: 'glove.42B.300d.txt'

In [63]:

```

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

```

Train Essays

In [64]:

```

# average Word2Vec
# compute average word2vec for each review.
avg_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
    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%|██████████| 8978/8978 [00:02<00:00, 3027.03it/s]

8978

300

Test Essay

In [65]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_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
    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%|██████████| 6600/6600 [00:02<00:00, 2877.56it/s]

6600

300

CV Essay

In [66]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_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
    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%|██████████| 4422/4422 [00:01<00:00, 2849.38it/s]

4422

300

Train Titles

In [67]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(preprocessed_titles_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_titles_train.append(vector)

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

100%|██████████| 8978/8978 [00:00<00:00, 56440.30it/s]

8978

300

Test Titles

In [68]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(preprocessed_titles_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_titles_test.append(vector)

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

100%|██████████| 6600/6600 [00:00<00:00, 56496.78it/s]

6600

300

CV Titles

In [69]:

```

# average Word2Vec
# compute average word2vec for each review.
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 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_titles_cv.append(vector)

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

```

100%|██████████| 4422/4422 [00:00<00:00, 55255.62it/s]

4422

300

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [70]:

```

# 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())

```

Train Essay

In [71]:

```

# 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
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split()))) # getting
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
            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%|██████████| 8978/8978 [00:17<00:00, 518.94it/s]

8978

300

Test Essay

In [72]:

```

# average Word2Vec
# 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()))) # getting
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
            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%|██████████| 6600/6600 [00:12<00:00, 560.73it/s]

6600

300

CV Essay

In [73]:

```
# 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
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split()))) # getting
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
            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%|██████████| 4422/4422 [00:08<00:00, 541.22it/s]

4422

300

In [74]:

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

Title Train

In [75]:

```

# 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()))) # getting
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
            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%|██████████| 8978/8978 [00:00<00:00, 21485.99it/s]

8978

300

Title Test

In [76]:

```

# compute average word2vec for each review.

tfidf_w2v_vectors_titles_test = [];

for sentence in tqdm(preprocessed_titles_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()))) # getting
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
            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%|██████████| 6600/6600 [00:00<00:00, 29774.26it/s]

6600

300

Title CV

In [77]:

```
# 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()))) # getting
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
            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%|██████████| 4422/4422 [00:00<00:00, 30186.10it/s]

4422

300

1.5.3 Vectorizing Numerical features : Price , Quantity ,teacher_number_of_previously_posted_projects

In [78]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index
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 [79]:

```
# check this one: https://www.youtube.com/watch?v=0H0qOcLn3Z4&t=530s
# standardization sklearn: https://scikitlearn.org/stable/modules/generated/sklearn.preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from sklearn import preprocessing
price_scalar = MinMaxScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standarddevi
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")
# Now standardize the data with above maen and variance.
price_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_train
# Now standardize the data with above maen and variance.
price_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
price_test
# Now standardize the data with above maen and variance.
price_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_cv
```

Out[79]:

```
array([[0.02033647],
       [0.00802916],
       [0.01633539],
       ...,
       [0.01690555],
       [0.0089164 ],
       [0.00033209]])
```

In [80]:

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

```
After vectorizations
(8978, 1) (8978,)
(4422, 1) (4422,)
(6600, 1) (6600,)
```

Quantity

In [81]:

```

price_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")
# Now standardize the data with above mean and variance.
quantity_train = price_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
quantity_train
# Now standardize the data with above mean and variance.
quantity_cv = price_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
quantity_cv
# Now standardize the data with above mean and variance.
quantity_test = price_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
print("After vectorizations")
print(quantity_train.reshape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)

```

After vectorizations

```

<built-in method reshape of numpy.ndarray object at 0x7fe3965ce710> (8978,)
(4422, 1) (4422,)
(6600, 1) (6600,)

```

In [82]:

quantity_train

Out[82]:

```

array([[0.13888889],
       [0.00793651],
       [0.04166667],
       ...,
       [0.          ],
       [0.01587302],
       [0.03571429]])

```

teacher_number_of_previously_posted_projects

In [83]:

```

price_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,
# print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}
# Now standardize the data with above mean and variance.
prev_projects_train = price_scalar.transform(X_train['teacher_number_of_previously_posted_projects'])
prev_projects_train
# Now standardize the data with above mean and variance.
prev_projects_cv = price_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'])
prev_projects_cv
# Now standardize the data with above mean and variance.
prev_projects_test = price_scalar.transform(X_test['teacher_number_of_previously_posted_projects'])
prev_projects_test

```

Out[83]:

```

array([[0.02457002],
       [0.01965602],
       [0.01965602],
       ...,
       [0.002457  ],
       [0.         ],
       [0.004914  ]])

```

In [84]:

```

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)

```

```

After vectorizations
(8978, 1) (8978,)
(4422, 1) (4422,)
(6600, 1) (6600,)

```

Title Word Count

In [85]:

```
price_scalar.fit(X_train['title_word_count'].values.reshape(-1,1)) # finding the mean and s
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}
# Now standardize the data with above maen and variance.
title_word_count_train = price_scalar.transform(X_train['title_word_count'].values.reshape(
title_word_count_train
# Now standardize the data with above maen and variance.
title_word_count_cv = price_scalar.transform(X_cv['title_word_count'].values.reshape(-1, 1)
title_word_count_cv
# Now standardize the data with above maen and variance.
title_word_count_test = price_scalar.transform(X_test['title_word_count'].values.reshape(-1
title_word_count_test
```

Out[85]:

```
array([[0.7],
       [0.3],
       [0.1],
       ...,
       [0.1],
       [0.2],
       [0.2]])
```

In [86]:

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

```
After vectorizations
(8978, 1) (8978,)
(4422, 1) (4422,)
(6600, 1) (6600,)
```

essay_word_count

In [87]:

```
price_scalar.fit(X_train['essay_word_count'].values.reshape(-1,1)) # finding the mean and s
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}
# Now standardize the data with above maen and variance.
essay_word_count_train = price_scalar.transform(X_train['essay_word_count'].values.reshape(-1,1))
essay_word_count_train
# Now standardize the data with above maen and variance.
essay_word_count_cv = price_scalar.transform(X_cv['essay_word_count'].values.reshape(-1, 1))
essay_word_count_cv
# Now standardize the data with above maen and variance.
essay_word_count_test = price_scalar.transform(X_test['essay_word_count'].values.reshape(-1, 1))
essay_word_count_test

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

After vectorizations

```
(8978, 1) (8978,)
(4422, 1) (4422,)
(6600, 1) (6600,)
```

1.5.4 Merging all the above features

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

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](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
- Find the best hyper paramter using k-fold cross validation (or) simple cross validation data
- Use gridsearch-cv or randomsearch-cv or 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](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) (<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/>) with predicted and original labels of test data points



4. [Task-2]

- Select top 2000 features from feature **Set 2** using `SelectKBest` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html) 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)
```

- Repeat the steps 2 and 3 on the data matrix after feature selection

#### 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](http://zetcode.com/python/prettytable/) (<http://zetcode.com/python/prettytable/>)



#### 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-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method `fit_transform()` on your train data, and apply the method `transform()` on cv/test data.
4. For more details please go through this [link](https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf). (<https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf>)

## 2. K Nearest Neighbor

In [88]:

```
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_categor
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categor
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_or
```

## Set 1: categorical, numerical features + project\_title(BOW) + preprocessed\_essay (BOW)

In [89]:

```
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)
```

```
Final Data matrix
(8978, 6445) (8978,)
(4422, 6445) (4422,)
(6600, 6445) (6600,)
```

## Finding hyperparameter for maximum value for AUC

In [90]:

```
def batch_predict(clf, data):
 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
 # 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 until 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
```

## Gridsearch Cv

In [91]:

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

neigh = KNeighborsClassifier(n_jobs=-1)

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

clf = GridSearchCV(neigh, parameters, cv= 2, scoring='roc_auc', return_train_score=True)

clf.fit(X_tr, y_train)

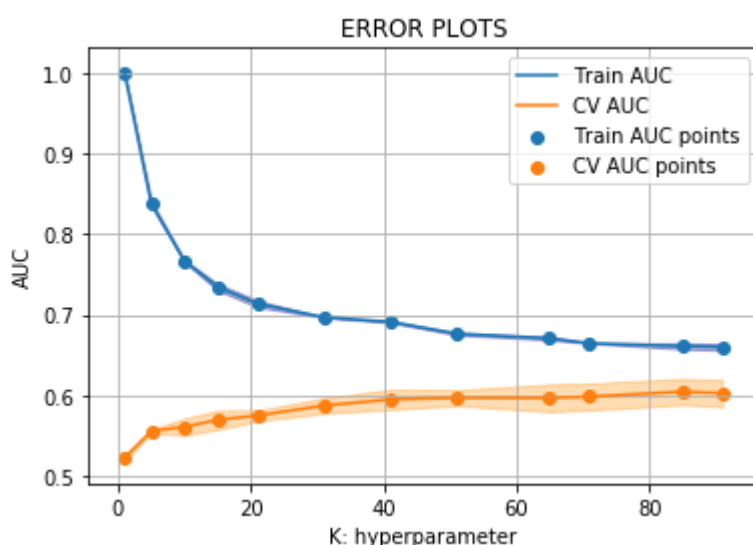
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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)

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,al

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()
```



## Training Model using best hyperParameter Value



In [92]:

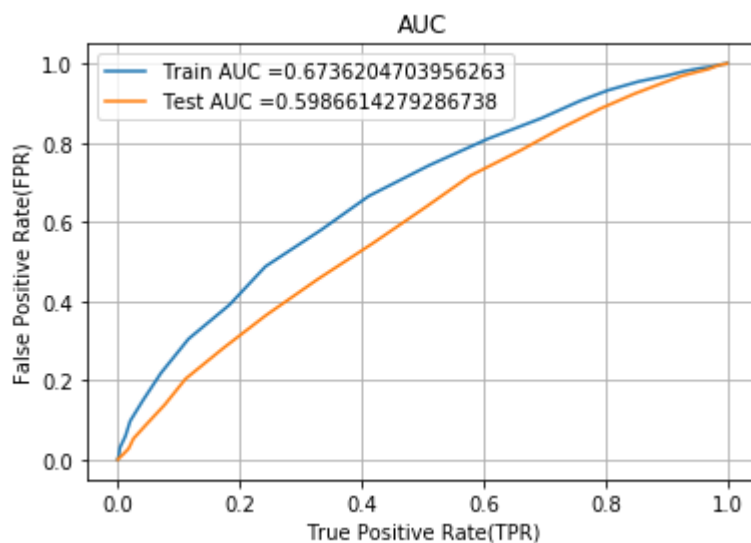
```
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.
from sklearn.metrics import roc_curve, auc

neigh = KNeighborsClassifier(n_neighbors=85)
neigh.fit(X_tr, y_train)
roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
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 [93]:

```
def predict(proba, threshold, fpr, tpr):

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

## Confusion matrix

In [94]:

```
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  $tpr \cdot (1 - fpr)$  0.24988781646827568 for threshold 0.8

```
[[670 699]
 [1956 5653]]
```

In [95]:

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresh
```

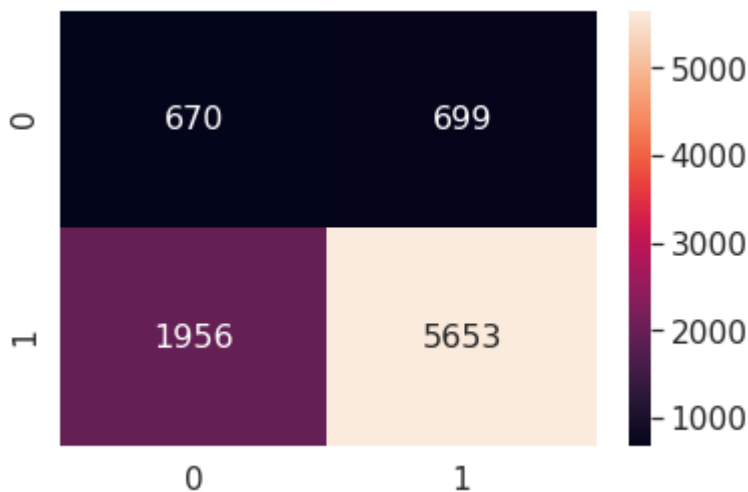
the maximum value of  $tpr \cdot (1 - fpr)$  0.24988781646827568 for threshold 0.8

In [96]:

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

Out[96]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe39616e470>



## Test Data

In [97]:

```
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  $tpr \cdot (1 - fpr)$  0.24997529732143917 for threshold 0.812

```
[[498 508]
 [2028 3566]]
```

In [98]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_threshold
```

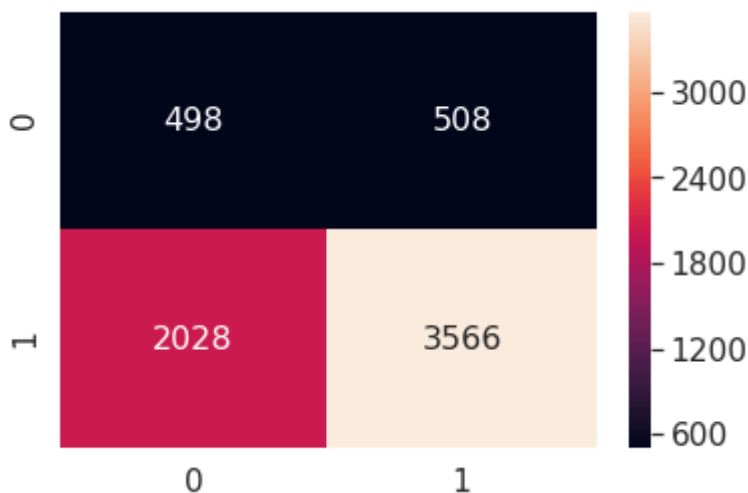
the maximum value of  $tpr \cdot (1 - fpr)$  0.24997529732143917 for threshold 0.812

In [99]:

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

Out[99]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe395f21fd0>



## Set 2 : categorical, numerical features + project\_title(TFIDF) + preprocessed\_essay (TFIDF)

In [100]:

```
Please write all the code with proper documentation
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_categor
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categorie
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_or

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
(8978, 6445) (8978,)
(4422, 6445) (4422,)
(6600, 6445) (6600,)
```

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

# Gridserach Cv

In [101]:

```
neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1, 5, 10, 15, 21, 33, 41, 51, 65, 71, 85, 91]}
clf = GridSearchCV(neigh, parameters, cv=2, scoring='roc_auc',return_train_score=True)
clf.fit(X_tr, y_train)

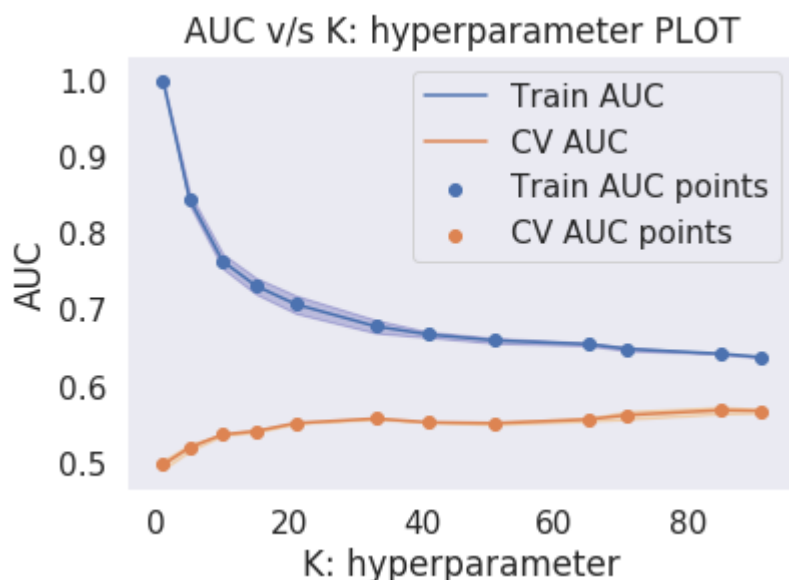
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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)

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,al

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")
plt.grid()
plt.show()
```



## Training Model using best hyperParameter Value

In [102]:

```

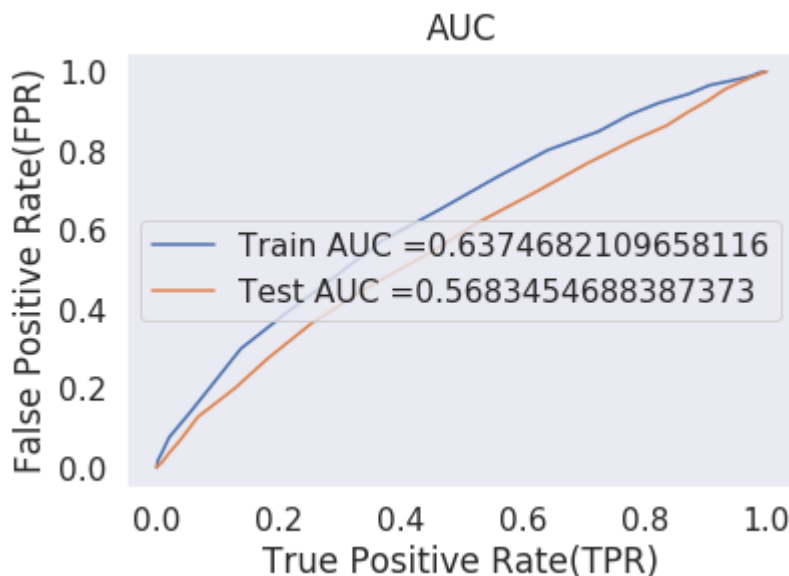
neigh = KNeighborsClassifier(n_neighbors=85)
neigh.fit(X_tr, y_train)
roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
not the predicted outputs

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()

```



## Confusion Matrix Train data

In [103]:

```

print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))

```

Train confusion matrix  
the maximum value of  $tpr \cdot (1 - fpr)$  0.24889537238262885 for threshold 0.847  
[[ 730 639]  
 [2634 4975]]

In [104]:

```
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

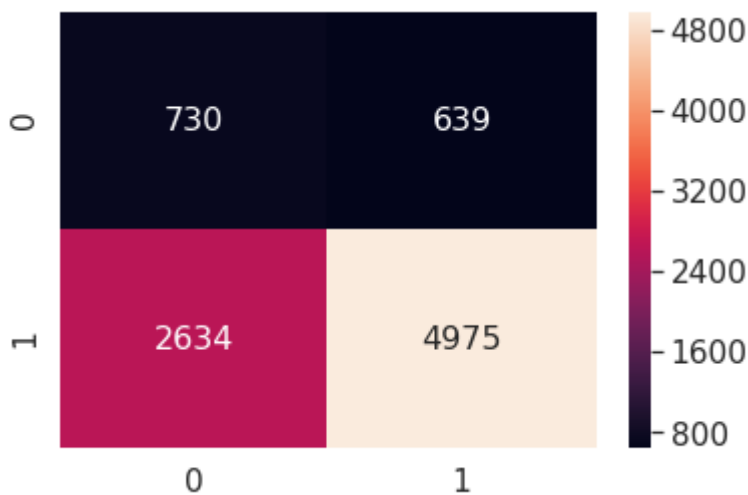
the maximum value of  $tpr \cdot (1-fpr)$  0.24889537238262885 for threshold 0.847

In [105]:

```
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[105]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe395f13240>



## Test Data

In [106]:

```
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  $tpr \cdot (1-fpr)$  0.24898817828614792 for threshold 0.859

```
[[558 448]
 [2571 3023]]
```

In [107]:

```
conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_threshc
```

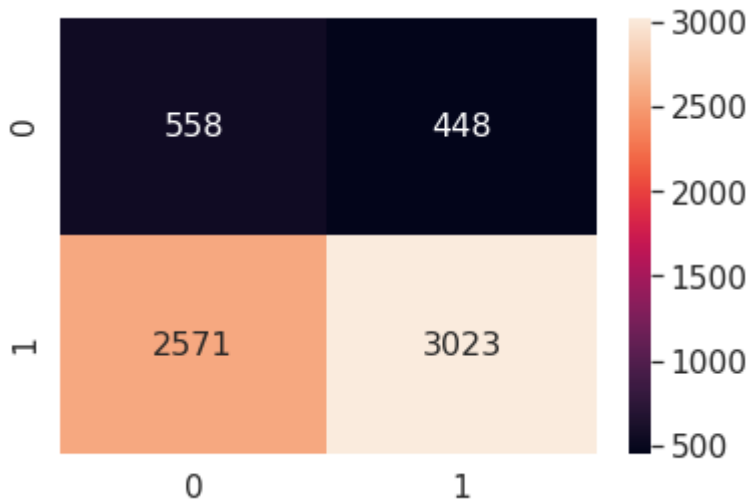
the maximum value of  $tpr \cdot (1-fpr)$  0.24898817828614792 for threshold 0.859

In [108]:

```
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[108]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe395eeceb8>



## Set 3 : categorical, numerical features + project\_title(AVG W2V) + preprocessed\_essay (AVG W2V)

In [109]:

```
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_categor
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categor
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_or

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)
```

```
Final Data matrix
(8978, 705) (8978,)
(4422, 705) (4422,)
(6600, 705) (6600,)
```

## Finding hyperparameter for maximum value for AUC

### GridSearch CV

In [110]:

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

neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1, 5, 10, 15, 21, 33, 41, 51, 65, 71, 85, 91]}
clf = GridSearchCV(neigh, parameters, cv=2, scoring='roc_auc',return_train_score=True)
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']

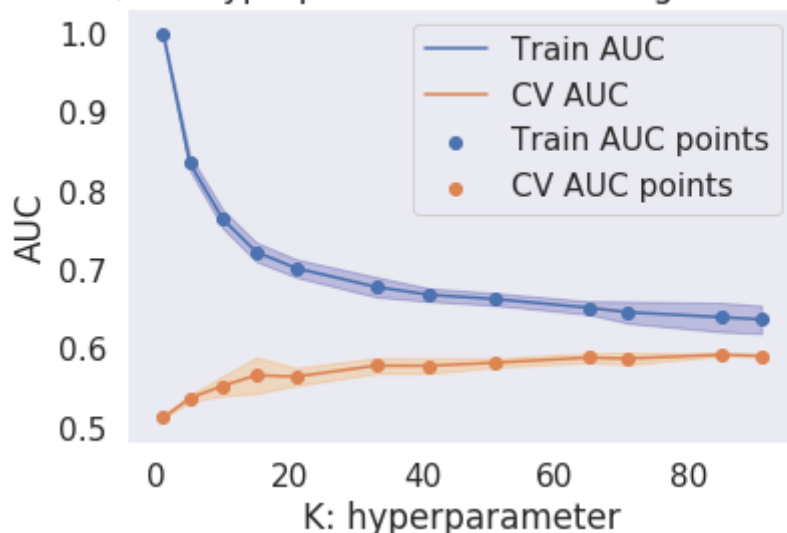
plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc + train_auc_std)

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,al

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()
```

AUC v/s K: hyperparameter Plot - using GridSearchcv



## Training Model using best hyperParameter Value



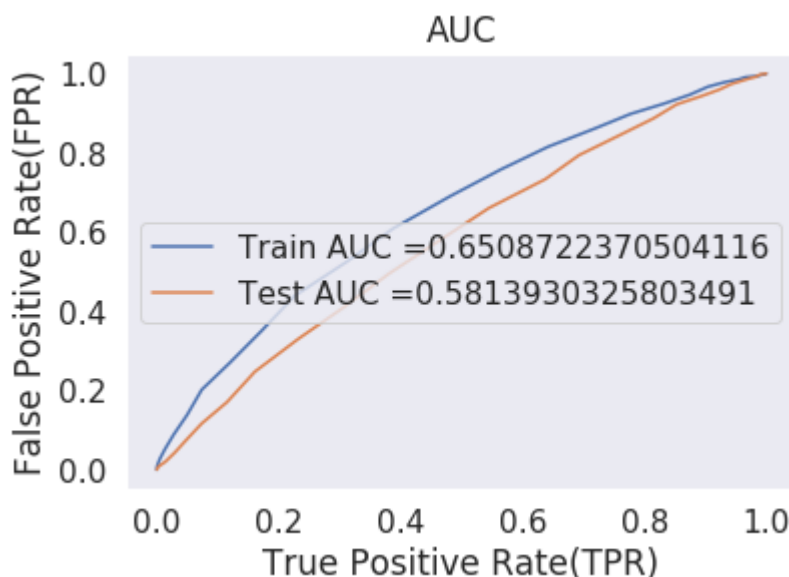
In [111]:

```
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.
neigh = KNeighborsClassifier(n_neighbors=91)
neigh.fit(X_tr, y_train)
roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
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()
```



## Confusion Data matrix :Train data

In [112]:

```
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
```

Train confusion matrix  
the maximum value of  $tpr \cdot (1 - fpr)$  0.24959648610765034 for threshold 0.835  
[[ 712 657]  
 [2365 5244]]

In [113]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

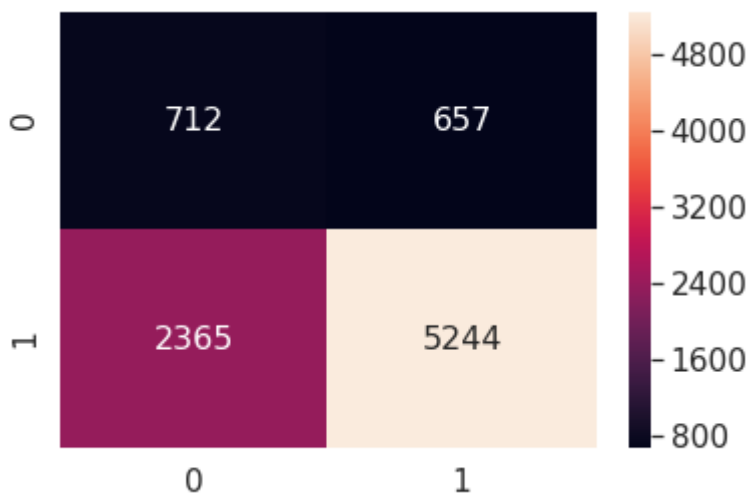
the maximum value of  $tpr \cdot (1-fpr)$  0.24959648610765034 for threshold 0.835

In [114]:

```
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[114]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe39606f320>



## Test Data

In [115]:

```
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  $tpr \cdot (1-fpr)$  0.24871941314340595 for threshold 0.846

```
[[539 467]
 [2355 3239]]
```

In [116]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_threshc
```

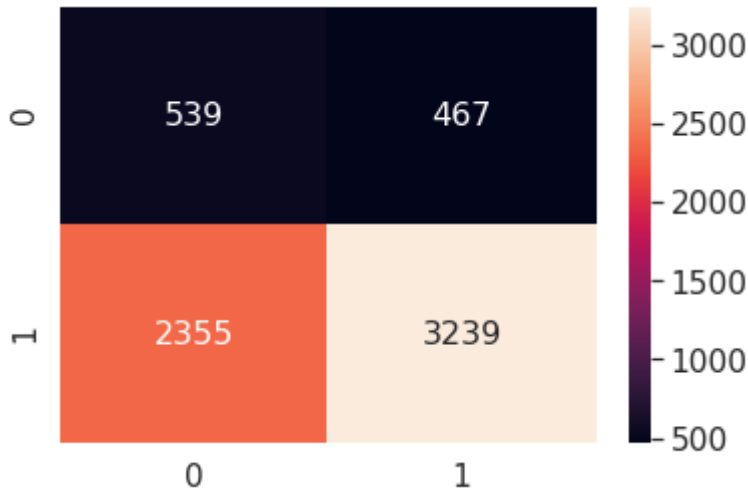
the maximum value of  $tpr \cdot (1-fpr)$  0.24871941314340595 for threshold 0.846

In [117]:

```
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[117]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe395e90da0>



## Set 4 : categorical, numerical features + project\_title(TFIDF W2V) + preprocessed\_essay (TFIDF W2V)

In [118]:

```
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_categor
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categorie
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_or

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)
```

```
Final Data matrix
(8978, 705) (8978,)
(4422, 705) (4422,)
(6600, 705) (6600,)
```

## GridSearch CV

In [119]:

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

neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1, 5, 10, 15, 21, 33, 41, 51, 65, 71, 85, 91]}
clf = GridSearchCV(neigh, parameters, cv=2, scoring='roc_auc',return_train_score=True)
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']

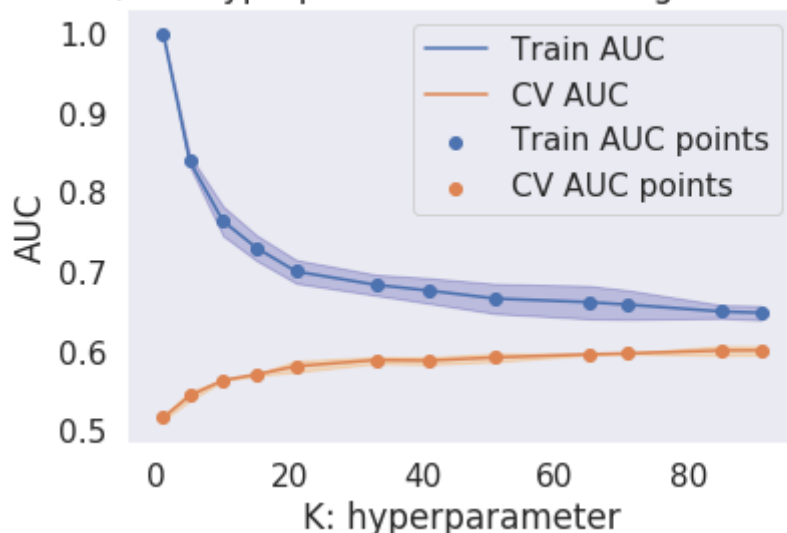
plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc + train_auc_std)

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,al

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()
```

AUC v/s K: hyperparameter Plot - using GridSearchcv



## Training Model using best hyperParameter Value

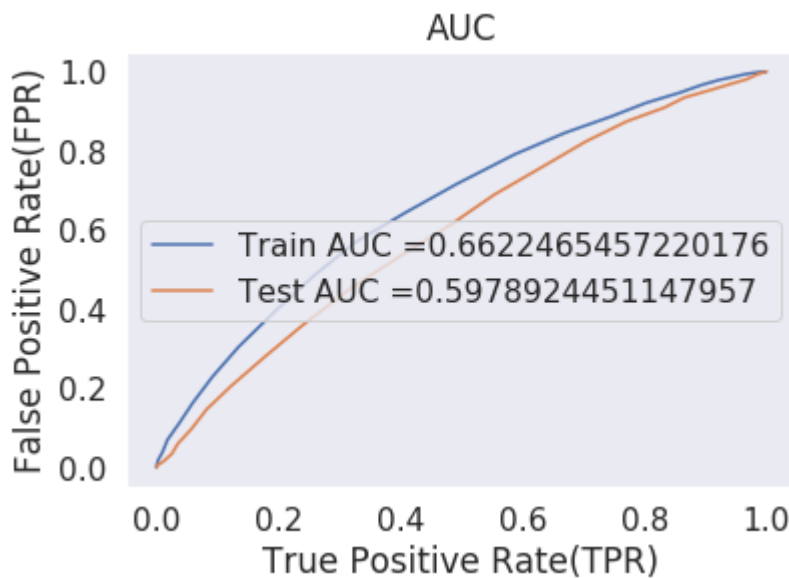
In [120]:

```
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.
neigh = KNeighborsClassifier(n_neighbors=91)
neigh.fit(X_tr, y_train)
roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
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()
```



## Confusion Data matrix :Train data

In [121]:

```
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
```

Train confusion matrix  
the maximum value of  $tpr \cdot (1 - fpr)$  0.24994117367718144 for threshold 0.824  
[[ 695 674]  
 [2158 5451]]

In [122]:

```
conf_matr_df_train_3 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

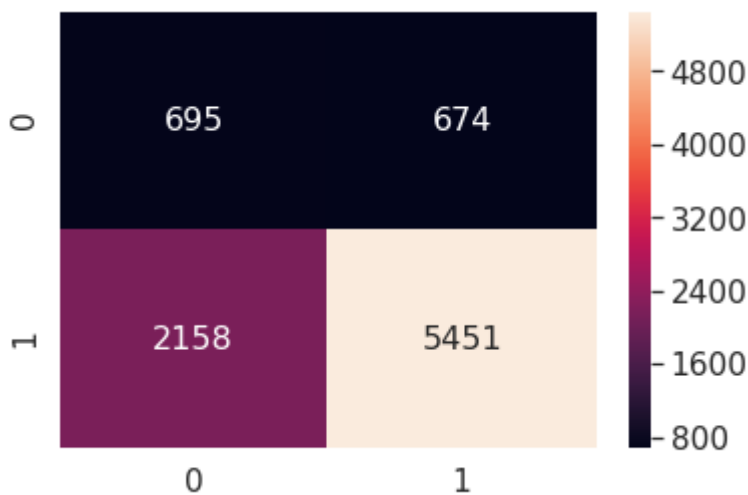
the maximum value of  $tpr \cdot (1 - fpr)$  0.24994117367718144 for threshold 0.824

In [123]:

```
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[123]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe386db9eb8>



## Test Data

In [125]:

```
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  $tpr \cdot (1 - fpr)$  0.24974704457153699 for threshold 0.846

```
[[607 399]
 [2622 2972]]
```

In [126]:

```
conf_matr_df_test_3 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresho
```

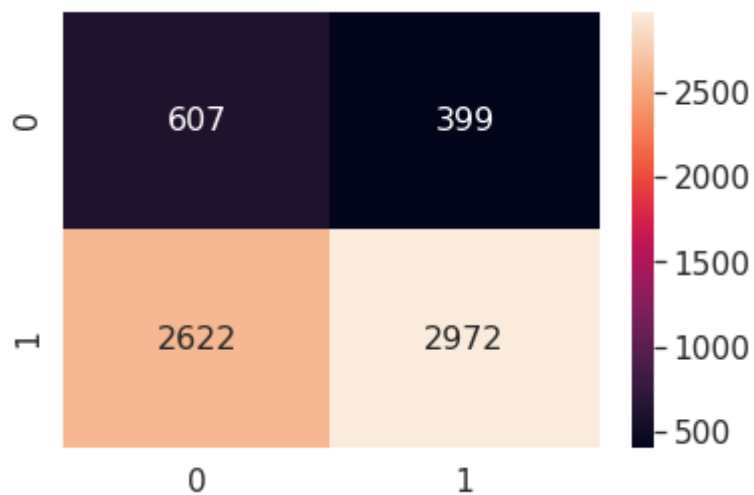
the maximum value of  $tpr \cdot (1 - fpr)$  0.24974704457153699 for threshold 0.846

In [127]:

```
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[127]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe395f84ef0>



## 2.5 Feature selection with `SelectKBest`

In [134]:

```

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_categories_one_hot_train))
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categories_one_hot_test))
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_one_hot_cv))

from sklearn.feature_selection import SelectKBest, chi2

selectk = SelectKBest(chi2 , k=2000).fit(X_tr,y_train)

X_tr_new = selectk.transform(X_tr)
X_te_new = selectk.transform(X_te)
X_cr_new = selectk.transform(X_cr)

print("Final Data matrix")
print(X_tr_new.shape, y_train.shape)
print(X_cr_new.shape, y_cv.shape)
print(X_te_new.shape, y_test.shape)

```

```

Final Data matrix
(8978, 2000) (8978,)
(4422, 2000) (4422,)
(6600, 2000) (6600,)

```

## GridSearch CV



In [137]:

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

neigh = KNeighborsClassifier(n_jobs=-1)

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

clf = GridSearchCV(neigh, parameters, cv=2, scoring='roc_auc',return_train_score=True)
clf.fit(X_tr_new, y_train)

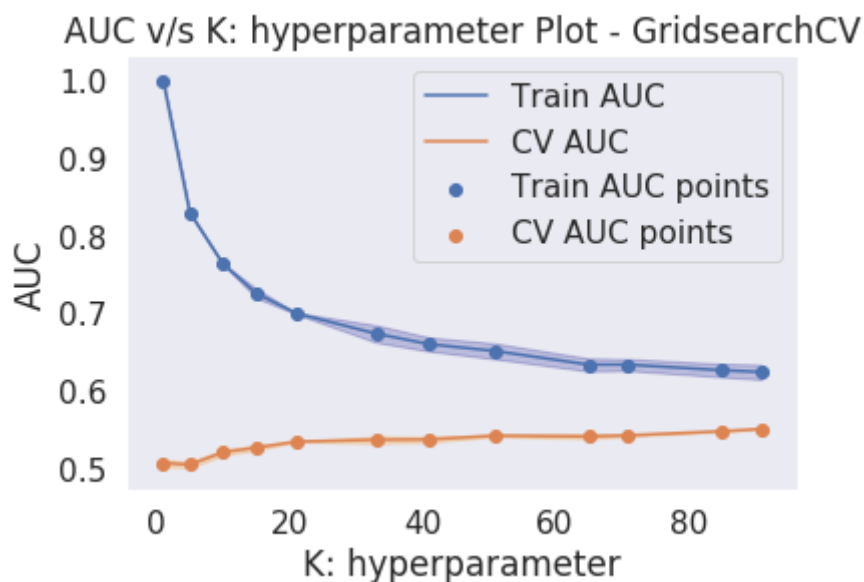
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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)

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,al

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()
```



## Training Model using best hyperParameter Value

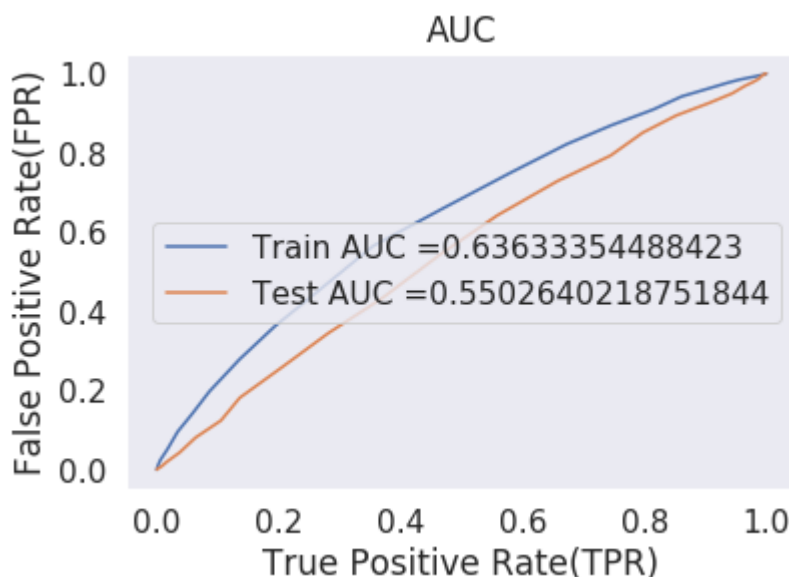
In [138]:

```
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.
neigh = KNeighborsClassifier(n_neighbors=91)
neigh.fit(X_tr_new, y_train)
roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
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()
```



## Confusion matrix Train data

In [139]:

```
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24928914858435325 for threshold 0.846
[[721 648]
 [2566 5043]]
```

In [140]:

```
conf_matr_df_train_4 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

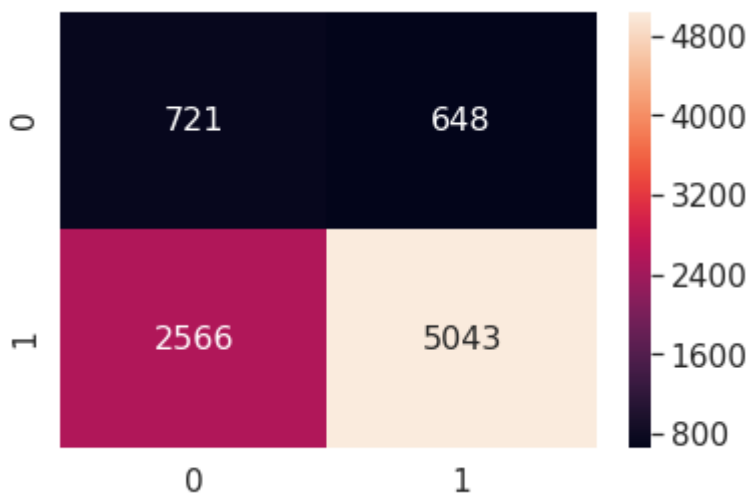
the maximum value of  $tpr*(1-fpr)$  0.24928914858435325 for threshold 0.846

In [141]:

```
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[141]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe395fc5e10>



## Test Data

In [142]:

```
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  $tpr*(1-fpr)$  0.24871941314340595 for threshold 0.868

```
[[633 373]
 [3167 2427]]
```

In [143]:

```
conf_matr_df_test_4 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_threshc
```

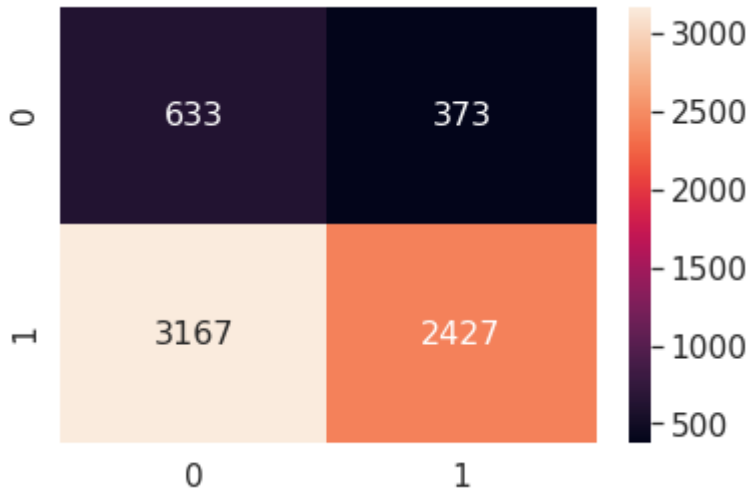
the maximum value of  $tpr*(1-fpr)$  0.24871941314340595 for threshold 0.868

In [144]:

```
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[144]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fe395fa84a8>



### 3. Conclusions

In [145]:

```
Please compare all your models using Prettytable Library
Compare all your models using Prettytable Library
http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "Train AUC", "Test AUC"]

x.add_row(["BOW", "Brute", 85, 0.67, 0.59])
x.add_row(["TFIDF", "Brute", 85, 0.63, 0.56])
x.add_row(["AVG W2V", "Brute", 91, 0.65, 0.58])
x.add_row(["TFIDF W2V", "Brute", 91, 0.66, 0.59])
x.add_row(["TFIDF", "Top 2000", 91, 0.63, 0.55])

print(x)
```

| Vectorizer | Model    | Hyper Parameter | Train AUC | Test AUC |
|------------|----------|-----------------|-----------|----------|
| BOW        | Brute    | 85              | 0.67      | 0.59     |
| TFIDF      | Brute    | 85              | 0.63      | 0.56     |
| AVG W2V    | Brute    | 91              | 0.65      | 0.58     |
| TFIDF W2V  | Brute    | 91              | 0.66      | 0.59     |
| TFIDF      | Top 2000 | 91              | 0.63      | 0.55     |

In [ ]: