DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics`

	• Literacy & Language • Math & Science • Music & The Arts • Special Needs • Warmth **Examples:** • Music & The Arts • Literacy & Language, Math & Science
school_state	State where school is located (Two-letter U.S. postal code). **Example:** WY
project_subject_subcategories	One or more (comma-separated) subject subcategories for the project. **Examples:** • Literacy • Literature & Writing, Social Sciences
project_resource_summary	An explanation of the resources needed for the project. **Example:** • My students need hands on literacy materials to manage sensory needs!
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2016-04-28 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project. **Example:** bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated values: • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same teacher. **Example:**

	2

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

```
F:\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detecte
d Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

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

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix'
'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:
```
# how to replace elements in list python: https://stackoverflow.com/a/258216
3/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(proje
ct_data.columns)]
```

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


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


project_data.head(2)
```

Out[4]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_stat
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

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

1.2 Preprocessing of `project_subject_categories`

In [6]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
```

```
    for j in i.split(','): # it will split it in three parts ["Math & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based
 on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going
 to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(e
mpty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the t
railing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 Preprocessing of `project_subject_subcategories`

In [7]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverfl
ow.com/a/47301924/4084039

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunge
r"
    for j in i.split(','): # it will split it in three parts ["Math & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based
 on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going
 to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(e
mpty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the t
railing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898
595/4084039
my_counter = Counter()
```

```
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.3 Text Preprocessing

Essay

In [8]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +project_da
ta["project_essay_2"].map(str) +project_data["project_essay_3"].map(str) +pr
oject_data["project_essay_4"].map(str)
```

In [9]:
```
project_data.head(2)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_stat
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

Train And Test Data Split :

In [10]:
```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(project_data,project_dat
a['project_is_approved'],stratify=project_data['project_is_approved'],test_s
ize=0.33)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,stratify=y_
train,test_size=0.33)


print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
```

```
(49041, 18) (49041,)
```

```
(24155, 18) (24155,)
(36052, 18) (36052,)
```

In [11]:
```
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 [12]:
```
# 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[20000])
print("="*50)
```

```
Excited, creative, ready to learn and explore - that's my group of Kinder
gartners! My students come to class ready to experience new things and to
learn as much as they can! \r\n\r\nI teach at a school with a diverse pop
ulation, including many students receiving free lunches and from families
that move a lot.\r\nIn my classroom, it's my goal that they find a welcom
ing environment where they feel free to learn, explore and always feel ac
cepted just as they are.Kids eat a lot, and that's a good thing. Kids nee
d energy to help them stay focused and alert during the day - especially
when they are in school! The amount of focus and brain power required by
kids in an 8 hour day is daunting sometimes. \r\n\r\nMany kids don't get
the food they require, many missing out on breakfast or eating foods that
don't give them any energy that lasts.\r\n\r\nWe supplement this by provi
ding snacks in our classroom, most of which are provided by parents and m
yself. But this isn't usually enough, and sometimes its snacks that don't
help the students in the long run. I want to be able to supplement those
snacks with other varieties so they are getting more food that helps them
keep their energy up and brains on!nannan
==================================================
Students walk in with smiles on their faces and excitement in their heart
s as they are ready for the day before them! These students come with a v
ariety of different backgrounds, as our school is open to anyone in the c
ounty. Our charter school was founded on the motto that it is a place whe
re \"teachers can teach and students will learn.\" The second graders who
will benefit from these materials are an exceptional group of learners an
d would love the opportunity to express themselves any chance they get!Wr
iting is a special art that students must strengthen at a young age, and
it is one that they will carry with them for the rest of their lives. You
ng children love to express themselves, these journals will give the oppo
rtunity to record, illustrate, and showcase their wonderful work.\r\n\r\n
Students will each have their own writing journal where they can write ab
out their favorite things. They will have a place to draw pictures to ill
ustrate the stories in their journals. The colored pencils are almost mag
ic in the sense that when water is painted on top of the colored pencils,
they look like watercolor paintings. \"Empowering Students Through Art\"
allows students to explore the many sides of art!nannan
==================================================
I teach kindergarten through fifth grade.  All of my students are in my c
lassroom due to a hearing loss. This creates a great difficulty in their
learning, understanding abstract concepts, and vocabulary related to acad
emics. They typically have a limited vocabulary and this interferes with
their ability to comprehend. The average deaf student graduates high scho
ol with a 4th grade reading level.\r\n\r\nEarly intervention is critical
for my students, as well as having access to appropriate curriculum and m
aterials. \r\n\r\nThey also need multiple opportunities for practice of l
```

```
earned skills, most of the district curriculum is not accessible to our s
tudents due to the high level of vocabulary and content. Ipads will benef
it our classroom by providing every student access to individualized onli
ne materials at their level.In addition to daily classroom instruction, w
e will use the iPads in my classroom during our math, science, and social
studies classes in order to provide opportunities for individual practice
for each students individual instruction. Additionally, the students will
use the iPads to look up signs for unknown vocabulary and visual represen
tation for words and concepts that are unfamiliar.\r\n\r\nWith the wide r
ange of language skills in my classroom, individualization is mandatory t
o reach each student at his or her current level.\r\n\r\nIPads will provi
de each student opportunities to practice at their individual level. We a
ppreciate any assistance that you can provide to help us attain our goal
s!nannan
==================================================
My awesome students come to my class everyday greeted at the door with a
hug and a listening ear! I want the children to have choices in my classr
oom and realize that they have ownership in their learning space. My clas
sroom is a happy place where I encourage the kids to be comfortable so th
ey can enjoy learning. I want every child to know that my classroom is a
safe place where they are loved.  I want learning to be a fun process and
for school to be a happy place to come everyday!  \r\nThe children love t
o sit on anything other than the standard chair.  Any chance they get to
sit in a different chair, especially the one like the wobble chair, they
love the experience.  They are comfortable and at the same time they are
exercising.  \r\n\r\nThey have to utilize their core as well as their bac
k to maintain good balance on the chair.  They love the challenge and the
y don't even realize they are exercising!  :)\r\n\r\nThe wobble chairs ar
e even more comfortable than our standard chairs we have in the classroom
now.  The children love to have a different option.nannan
==================================================
```

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

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

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

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

```
My awesome students come to my class everyday greeted at the door with a
hug and a listening ear! I want the children to have choices in my classr
oom and realize that they have ownership in their learning space. My clas
sroom is a happy place where I encourage the kids to be comfortable so th
ey can enjoy learning. I want every child to know that my classroom is a
safe place where they are loved.  I want learning to be a fun process and
```

```
for school to be a happy place to come everyday!  \r\nThe children love t
o sit on anything other than the standard chair.  Any chance they get to
sit in a different chair, especially the one like the wobble chair, they
love the experience.  They are comfortable and at the same time they are
exercising.  \r\n\r\nThey have to utilize their core as well as their bac
k to maintain good balance on the chair.  They love the challenge and the
y do not even realize they are exercising!  :)\r\n\r\nThe wobble chairs a
re even more comfortable than our standard chairs we have in the classroo
m now.  The children love to have a different option.nannan
==================================================
```

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

```
My awesome students come to my class everyday greeted at the door with a
hug and a listening ear! I want the children to have choices in my classr
oom and realize that they have ownership in their learning space. My clas
sroom is a happy place where I encourage the kids to be comfortable so th
ey can enjoy learning. I want every child to know that my classroom is a
safe place where they are loved.  I want learning to be a fun process and
for school to be a happy place to come everyday!    The children love to
sit on anything other than the standard chair.  Any chance they get to si
t in a different chair, especially the one like the wobble chair, they lo
ve the experience.  They are comfortable and at the same time they are ex
ercising.      They have to utilize their core as well as their back to m
aintain good balance on the chair.  They love the challenge and they do n
ot even realize they are exercising!  :)    The wobble chairs are even mo
re comfortable than our standard chairs we have in the classroom now.  Th
e children love to have a different option.nannan
```

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

```
My awesome students come to my class everyday greeted at the door with a
hug and a listening ear I want the children to have choices in my classro
om and realize that they have ownership in their learning space My classr
oom is a happy place where I encourage the kids to be comfortable so they
can enjoy learning I want every child to know that my classroom is a safe
place where they are loved I want learning to be a fun process and for sc
hool to be a happy place to come everyday The children love to sit on any
thing other than the standard chair Any chance they get to sit in a diffe
rent chair especially the one like the wobble chair they love the experie
nce They are comfortable and at the same time they are exercising They ha
ve to utilize their core as well as their back to maintain good balance o
n the chair They love the challenge and they do not even realize they are
exercising The wobble chairs are even more comfortable than our standard
chairs we have in the classroom now The children love to have a different
option nannan
```

In [17]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'yo
u', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'h
e', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
```

```
'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'bec
ause', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into'
, 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 't
han', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shou
ld've", 'now', 'd', 'll', 'm', 'o', 're', \
            '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',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "sho
uldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_train.append(sent.lower().strip())
```

```
100%|██████████| 49041/49041 [00:23<00:00, 2067.20it/s]
```

In [19]:
```
# after preprocesing
preprocessed_essays_train[10000]
```

Out[19]:
```
'classroom students readers mathematicians scientists artists historians
outside school talents interests include dance gymnastics basketball socc
er arts crafts name classroom library center learning students explore sp
ace discover favorite authors fall love new series get inspired american
heroes learn past become super smart nonfiction topics much community lea
ders reading workshop consists mini lesson approximately 30 minutes indep
endent partner reading time student book box contains 5 10 books times st
udents encouraged book shop day fill boxes high quality high interest rig
ht books need variety genres levels topics choose keep engaged eager read
selection reflects students personal interests reading levels balance fic
tion nonfiction wide range subjects students expand horizons explore new
people places books nannan'
```

Preprocessing on Essay In Test Data :

```
```

In [20]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_test = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_test.append(sent.lower().strip())
```

```
100%|██████████| 36052/36052 [00:17<00:00, 2095.28it/s]
```

In [21]:
```
preprocessed_essays_test[10000]
```

Out[21]:
```
'classroom 22 wonderful young minds busy bodies school located high pover
ty community resources fairly limited many educational barriers students
work hard create maintain positive environment students successful suppor
t changes enhance students school experiences school building outdoor cla
ssroom enable students learn local natural environment still times learn
traditional classroom space sitting still traditional classroom seating e
xtreme challenge many children let us honest adults challenge fit born st
and dr seuss mindset also embrace challenged always recognize traditional
classroom flexible seating overcomes challenge every tradition classroom
faces embrace fact every child needs something different successful flexi
ble seating simple change make give students choice movement throughout d
ay increase success many plans restructuring classroom seating project fu
nded eliminate two tables classroom plan raise one remaining tables enabl
e students simply stand choose flexible seating resources receive extreme
ly important providing students multiple seating options along standing s
itting floor cushion choose three tables stools wobble seats traditional
chairs cushions balance ball chairs anticipate change cause uncomfortable
times eagerly awaiting sense relief many students feel not force bodies c
onform traditional seating plan flexible seating put control sit benefit
ways obvious seating options help develop core body strength balance othe
rs simply enable sit comfortable way reasons compelling students also buy
child comfortable likely engaged one not resources make huge positive imp
act students extremely excited choice sit nannan'
```

Preprocessing on Essay in CV Data :

In [22]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_cv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_cv.append(sent.lower().strip())
```

```
100%|██████████| 24155/24155 [00:11<00:00, 2116.09it/s]
```

In [23]:
```
preprocessed_essays_cv[10000]
```

Out[23]:
```
'easier build strong children repair broken men frederick douglas many st
udents need support within school environment help navigate way peer conf
licts stress management coping loss work students help strengthen social
emotional skills successful school students work learning regulate emotio
ns practicing ways positively interact peers working towards becoming bes
t self skills allow grow empathetic caring adults many students work requ
ire type sensory input getting need met better able attend class activity
lesson sensory items improve student ability learn focus within classroom
setting work students grades k 6 items dispersed students across school b
uilding based individual needs fidget toys help keep hands busy brain foc
used students need keep hands busy bodies moving order help focus improve
self regulation skills students access items right within classroom stude
nts also sensory breaks built daily schedule using visual timer helps tra
nsition break nannan'
```

Preprocessing on Project Title Train Data :

In [24]:
```
print(X_train['project_title'].values[0])
print("="*50)
print(X_train['project_title'].values[150])
print("="*50)
print(X_train['project_title'].values[1000])
print("="*50)
print(X_train['project_title'].values[20000])
print("="*50)
```

```
Feeding the Kinders
==================================================
Watercolors and Words
==================================================
iPads to Enhance the World of Learning!
==================================================
Wobble Chairs for My Classroom
==================================================
```

In [25]:
```
from tqdm import tqdm
preprocessed_projectitle_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e not in stopwords)
    preprocessed_projectitle_train.append(sent1.lower().strip())
```

```
100%|██████████| 49041/49041 [00:01<00:00, 47755.13it/s]
```

In [26]:
```
preprocessed_projectitle_train[5000]
```

Out[26]:
```
'music math class'
```

Preprocessing on Project Title CV Data :

In [27]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_projectitle_cv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e not in stopwords)
    preprocessed_projectitle_cv.append(sent1.lower().strip())
```

```
100%|██████████| 24155/24155 [00:00<00:00, 46792.27it/s]
```

In [28]:
```
# after preprocesing
preprocessed_projectitle_cv[19995:20000]
```

Out[28]:
```
['portable lab',
 'help us become readers',
 'touchscreen technology terrific third graders',
 'what shaking pt 3 jingle bells for the music room',
 'carpet easel miss edgar rockstar 1st graders']
```

Preprocessing on Project Title Test Data :

In [29]:
```
from tqdm import tqdm
preprocessed_projectitle_test = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e not in stopwords)
    preprocessed_projectitle_test.append(sent1.lower().strip())
```

```
100%|██████████| 36052/36052 [00:00<00:00, 40472.66it/s]
```

In [30]:
```
# after preprocesing
preprocessed_projectitle_test[19995:20000]
```

Out[30]:
```
['tablets learning',
 'join us our table',
 'science notebooks',
 'pre k kids need printer',
 'history technology']
```

DATA PREPROCESSING OF TEACHER_PREFIX IN TRAIN DATA :

In [31]:
```
from tqdm import tqdm
preprocessed_tf_train = []
# tqdm is for printing the status bar
```

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

```
100%|██████████| 49041/49041 [00:01<00:00, 34418.85it/s]
```

In [32]:
```
X_train['teacher_prefix'].fillna('',inplace=True)
```

DATA PREPROCESSING OF TEACHER_PREFIX IN CV DATA :

In [33]:
```
from tqdm import tqdm
preprocessed_tf_cv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['teacher_prefix'].values):
    sent = sent.replace('\\r', '')
    sent = sent.replace('\\"', '')
    sent = sent.replace('\\n', '')
    sent = re.sub('[^A-Za-z0-9]+', '', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_tf_cv.append(sent.lower().strip())
```

```
100%|██████████| 24155/24155 [00:00<00:00, 33565.82it/s]
```

In [34]:
```
X_cv['teacher_prefix'].fillna('',inplace=True)
```

DATA PREPROCESSING OF TEACHER_PREFIX IN TEST DATA :

In [35]:
```
from tqdm import tqdm
preprocessed_tf_test = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['teacher_prefix'].values):
    sent = sent.replace('\\r', '')
    sent = sent.replace('\\"', '')
    sent = sent.replace('\\n', '')
    sent = re.sub('[^A-Za-z0-9]+', '', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_tf_test.append(sent.lower().strip())
```

```
100%|██████████| 36052/36052 [00:01<00:00, 33576.05it/s]
```

In [36]:
```
X_test['teacher_prefix'].fillna('',inplace=True)
```

1.5 Preparing Data For Models

In [37]:
```
project_data.columns
```

Out[37]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_stat
e',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approv
ed',
       'clean_categories', 'clean_subcategories', 'essay'],
      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/

ONE HOT ENCODING OF CLEAN_CATAGORIES IN TRAIN,TEST,CV DATA :

In [38]:
```
# 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()), lowerc
ase=False, binary=True)
vectorizer.fit(X_train['clean_categories'].values)
categories_one_hot_train = vectorizer.transform(X_train['clean_categories'].
values)
categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].va
lues)
categories_one_hot_cv = vectorizer.transform(X_cv['clean_categories'].values
)
print(vectorizer.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",categories_one
_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",categories_one_
hot_test.shape)
print("Shape of matrix of CV data after one hot encoding ",categories_one_ho
t_cv.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix of Train data after one hot encoding  (49041, 9)
Shape of matrix of Test data after one hot encoding  (36052, 9)
Shape of matrix of CV data after one hot encoding  (24155, 9)
```

ONE HOT ENCODING OF CLEAN_SUB_CATAGORIES IN TRAIN DATA :

In [39]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lo
wercase=False, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values)
sub_categories_one_hot_train = vectorizer.transform(X_train['clean_subcatego
ries'].values)
sub_categories_one_hot_test = vectorizer.transform(X_test['clean_subcategori
es'].values)
sub_categories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories']
.values)
print(vectorizer.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",sub_categories
_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",sub_categories_
one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",sub
_categories_one_hot_cv.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutritio
nEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness',
'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics',
'Literacy']
Shape of matrix of Train data after one hot encoding  (49041, 30)
Shape of matrix of Test data after one hot encoding  (36052, 30)
Shape of matrix of Cross Validation data after one hot encoding  (24155,
30)
```

ONE HOT ENCODING OF SCHOOL STATE IN TEST,TRAIN,CV DATA :

In [40]:
```
from collections import Counter
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
ss_dict = dict(my_counter)
sorted_ss_dict = dict(sorted(ss_dict.items(), key=lambda kv: kv[1]))
```

In [41]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_ss_dict.keys()), lowerca
se=False, binary=True)
vectorizer.fit(X_train['school_state'].values)
school_state_categories_one_hot_train = vectorizer.transform(X_train['school
_state'].values)
```

```
school_state_categories_one_hot_test = vectorizer.transform(X_test['school_s
tate'].values)
school_state_categories_one_hot_cv = vectorizer.transform(X_cv['school_stat
e'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix of Train data after one hot encoding",school_state_ca
tegories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_ca
tegories_one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding",scho
ol_state_categories_one_hot_cv.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME',
'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS',
'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA',
'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL',
'NY', 'TX', 'CA']
Shape of matrix of Train data after one hot encoding (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of Cross Validation data after one hot encoding (24155, 5
1)
```

ONE HOT ENCODING OF TEACHER PREFIX IN TEST,TRAIN,CV DATA :

In [42]:
```
#Teacher Prefix
vectorizer = CountVectorizer(binary=True)
tp_one_hot=vectorizer.fit(X_train['teacher_prefix'].values)
teacher_prefix_categories_one_hot_train =vectorizer.transform(X_train['teach
er_prefix'].values)
teacher_prefix_categories_one_hot_test =vectorizer.transform(X_test['teacher
_prefix'].values)
teacher_prefix_categories_one_hot_cv=vectorizer.transform(X_cv['teacher_pref
ix'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_on
e_hot_train.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_on
e_hot_test.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_on
e_hot_cv.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encoding  (49041, 5)
Shape of matrix after one hot encoding  (36052, 5)
Shape of matrix after one hot encoding  (24155, 5)
```

ONE HOT ENCODING OF PROJECT GRADE CATAGORY IN TEST,TRAIN,CV DATA:

In [43]:
```
#Project Grade Catagory
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.split())
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
pgc_dict = dict(my_counter)
sorted_pgc_dict = dict(sorted(pgc_dict.items(), key=lambda kv: kv[1]))
```

In [44]:
```
del sorted_pgc_dict["Grades"]
```

In [45]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_pgc_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values)
print(vectorizer.get_feature_names())
pgc_one_hot_train=vectorizer.transform(X_train['project_grade_category'].values)
pgc_one_hot_cv=vectorizer.transform(X_cv['project_grade_category'].values)
pgc_one_hot_test=vectorizer.transform(X_test['project_grade_category'].values)
print("Shape of matrix after one hot encoding ",pgc_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",pgc_one_hot_cv.shape)
print("Shape of matrix after one hot encoding ",pgc_one_hot_test.shape)
```

```
['9-12', '6-8', '3-5', 'PreK-2']
Shape of matrix after one hot encoding  (49041, 4)
Shape of matrix after one hot encoding  (24155, 4)
Shape of matrix after one hot encoding  (36052, 4)
```

1.5.2 Vectorizing Text data

Bag of words on ESSAY in TRAIN , TEST AND CV Data :

In [46]:
```
# We are considering only the words which appeared in at least 10 documents (rows or projects).
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(preprocessed_essays_train)
text_bow_train = vectorizer.transform(preprocessed_essays_train)
text_bow_cv = vectorizer.transform(preprocessed_essays_cv)
text_bow_test = vectorizer.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_bow_train.shape)
print("Shape of matrix after one hot encoding ",text_bow_cv.shape)
print("Shape of matrix after one hot encoding ",text_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 11994)
Shape of matrix after one hot encoding  (24155, 11994)
Shape of matrix after one hot encoding  (36052, 11994)
```

Bag of words on TITLE in TRAIN , TEST AND CV Data :

In [47]:
```
vectorizer.fit(preprocessed_projectitle_train)
title_bow_train = vectorizer.transform(preprocessed_projectitle_train)
title_bow_cv = vectorizer.transform(preprocessed_projectitle_cv)
title_bow_test = vectorizer.transform(preprocessed_projectitle_test)
print("Shape of matrix after one hot encoding ",title_bow_train.shape)
print("Shape of matrix after one hot encoding ",title_bow_cv.shape)
print("Shape of matrix after one hot encoding ",title_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 2092)
Shape of matrix after one hot encoding  (24155, 2092)
Shape of matrix after one hot encoding  (36052, 2092)
```

1.5.2.2 TFIDF vectorizer on ESSAY in TRAIN , CV & TEST DATA :

In [48]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(preprocessed_essays_train)
text_tfidf_train = vectorizer.transform(preprocessed_essays_train)
text_tfidf_cv = vectorizer.transform(preprocessed_essays_cv)
text_tfidf_test = vectorizer.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_tfidf_train.shape)
print("Shape of matrix after one hot encoding ",text_tfidf_cv.shape)
print("Shape of matrix after one hot encoding ",text_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 11994)
Shape of matrix after one hot encoding  (24155, 11994)
Shape of matrix after one hot encoding  (36052, 11994)
```

TFIDF vectorizer on TITLE in TRAIN , CV & TEST DATA :

In [49]:
```
vectorizer.fit(preprocessed_projectitle_train)
title_tfidf_train = vectorizer.transform(preprocessed_projectitle_train)
title_tfidf_cv = vectorizer.transform(preprocessed_projectitle_cv)
title_tfidf_test = vectorizer.transform(preprocessed_projectitle_test)
print("Shape of matrix after one hot encoding ",title_tfidf_train.shape)
print("Shape of matrix after one hot encoding ",title_tfidf_cv.shape)
print("Shape of matrix after one hot encoding ",title_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 2092)
Shape of matrix after one hot encoding  (24155, 2092)
Shape of matrix after one hot encoding  (36052, 2092)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [50]:
```
# 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(' '))
for i in preprocessed_projectitle_train:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
```

```
words = set(words)
print("the unique words in the coupus", len(words))
inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our co
upus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")
words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))
# stronging variables into pickle files python: http://www.jessicayung.com/h
ow-to-use-pickle-to-save-and-load-variables-in-python/
import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)
```

```
Loading Glove Model
```

```
1917495it [03:35, 8887.20it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 6986826
the unique words in the coupus 42941
The number of words that are present in both glove vectors and our coupus
39194 ( 91.274 %)
word 2 vec length 39194
```

In [51]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/h
ow-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

AVG W2C on ESSAY IN TRAIN , CV & TEST DATA :

In [52]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_essay_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_essay_train.append(vector)

print(len(avg_w2v_vectors_essay_train))
print(len(avg_w2v_vectors_essay_train[0]))
```

```
100%|██████████| 49041/49041 [00:12<00:00, 3806.27it/s]
```

```
49041
300
```

In [53]:
```
avg_w2v_vectors_essay_cv = []; # the avg-w2v for each sentence/review is sto
red 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_essay_cv.append(vector)

print(len(avg_w2v_vectors_essay_cv))
print(len(avg_w2v_vectors_essay_cv[0]))
```

```
100%|██████████| 24155/24155 [00:08<00:00, 2690.51it/s]
```

```
24155
300
```

In [54]:
```
avg_w2v_vectors_essay_test = []; # the avg-w2v for each sentence/review is s
tored 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_essay_test.append(vector)

print(len(avg_w2v_vectors_essay_test))
print(len(avg_w2v_vectors_essay_test[0]))
```

```
100%|██████████| 36052/36052 [00:11<00:00, 3200.72it/s]
```

```
36052
300
```

AVG W2V on Project Title in Train , Test & CV Data :

In [55]:
```
avg_w2v_vectors_title_train = []; # the avg-w2v for each sentence/review is
 stored in this list
for sentence in tqdm(preprocessed_projectitle_train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_train.append(vector)
```

```
print(len(avg_w2v_vectors_title_train))
print(len(avg_w2v_vectors_title_train[0]))
```

```
100%|██████████| 49041/49041 [00:00<00:00, 74867.89it/s]
```

```
49041
300
```

In [56]:
```
avg_w2v_vectors_title_cv = []; # the avg-w2v for each sentence/review is sto
red in this list
for sentence in tqdm(preprocessed_projectitle_cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_cv.append(vector)

print(len(avg_w2v_vectors_title_cv))
print(len(avg_w2v_vectors_title_cv[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 52283.99it/s]
```

```
24155
300
```

In [57]:
```
avg_w2v_vectors_title_test = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm(preprocessed_projectitle_test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_test.append(vector)

print(len(avg_w2v_vectors_title_test))
print(len(avg_w2v_vectors_title_test[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 70673.74it/s]
```

```
36052
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V on ESSAY in TRAIN , CV & TEST Data :

In [58]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [59]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_essay_train = []; # the avg-w2v for each sentence/review i
s 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/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_train.append(vector)

print(len(tfidf_w2v_vectors_essay_train))
print(len(tfidf_w2v_vectors_essay_train[0]))
```

```
100%|██████████| 49041/49041 [01:20<00:00, 611.03it/s]
```

```
49041
300
```

In [60]:
```
tfidf_w2v_vectors_essay_cv = []; # the avg-w2v for each sentence/review is s
tored 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/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_cv.append(vector)

print(len(tfidf_w2v_vectors_essay_cv))
print(len(tfidf_w2v_vectors_essay_cv[0]))
```

```
100%|██████████| 24155/24155 [00:39<00:00, 614.02it/s]
```

```
24155
300
```

In [61]:
```
tfidf_w2v_vectors_essay_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/rev
iew
```

```
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_test.append(vector)

print(len(tfidf_w2v_vectors_essay_test))
print(len(tfidf_w2v_vectors_essay_test[0]))
```

```
100%|██████████| 36052/36052 [00:58<00:00, 613.17it/s]
```

```
36052
300
```

TFIDF weighted W2V on Project_Title in TRAIN , TEST & CV Data :

In [62]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_projectitle_train)
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [63]:
```
tfidf_w2v_vectors_title_train = []; # the avg-w2v for each sentence/review i
s stored in this list
for sentence in tqdm(preprocessed_projectitle_train): # for each review/sent
ence
    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/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_train.append(vector)

print(len(tfidf_w2v_vectors_title_train))
print(len(tfidf_w2v_vectors_title_train[0]))
```

```
100%|██████████| 49041/49041 [00:01<00:00, 38786.22it/s]
```

```
49041
300
```

In [64]:
```
tfidf_w2v_vectors_title_cv = []; # the avg-w2v for each sentence/review is s
```

```
tored in this list
for sentence in tqdm(preprocessed_projectitle_cv): # for each review/sentenc
e
    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/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_cv.append(vector)

print(len(tfidf_w2v_vectors_title_cv))
print(len(tfidf_w2v_vectors_title_cv[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 39190.57it/s]
```

```
24155
300
```

In [65]:
```
tfidf_w2v_vectors_title_test = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm(preprocessed_projectitle_test): # for each review/sente
nce
    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/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_test.append(vector)

print(len(tfidf_w2v_vectors_title_test))
print(len(tfidf_w2v_vectors_title_test[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 40327.64it/s]
```

```
36052
300
```

1.5.3 Vectorizing Numerical features

1) PRICE

```
```

In [67]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'su
m'}).reset_index()
price_data.head(5)
```

Out[67]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98
4	p000005	485.99	8

In [68]:
```
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
X_train.head()
```

Out[68]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
0	161274	p061673	bb723ed78077a86a0d3720380364369d	Mrs.	WA	
1	96842	p186128	2177c54ab7b5ef4e282979029c6a951c	Mrs.	SC	
2	142116	p160155	37652ef7e207601b2e0dbb115688e71e	Ms.	CA	
3	115651	p040421	3065a8cf3d7d96563ee3fb63bf3a9500	Mrs.	TX	
4	86217	p065997	4652fea0927b7a994477760bc924562c	Ms.	KY	

In [69]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler
price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
# Now standardize the data with above maen and variance.
price_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
```

In [70]:
```
print("The shape of training is",price_train.shape, y_train.shape)
print("The shape of cv is",price_cv.shape, y_cv.shape)
print("The shape of test is",price_test.shape, y_test.shape)
```

```
The shape of training is (49041, 1) (49041,)
The shape of cv is (24155, 1) (24155,)
The shape of test is (36052, 1) (36052,)
```

2) Quantity :

In [71]:
```
quantity_scalar = StandardScaler()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1))
quantity_train = quantity_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
quantity_cv = quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
quantity_test = quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
```

```
F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

In [72]:
```
print("The shape of training is",quantity_train.shape, y_train.shape)
print("The shape of cv is",quantity_cv.shape, y_cv.shape)
print("The shape of test is",quantity_test.shape, y_test.shape)
```

```
The shape of training is (49041, 1) (49041,)
The shape of cv is (24155, 1) (24155,)
The shape of test is (36052, 1) (36052,)
```

3) Number Of Projects Proposed By Teachers :

In [73]:
```
teacher_number_of_previously_posted_projects_scalar = StandardScaler()
teacher_number_of_previously_posted_projects_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
teacher_number_of_previously_posted_projects_train = teacher_number_of_previously_posted_projects_scalar.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_cv = teacher_number_of_previously_posted_projects_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_test = teacher_number_of_previously_posted_projects_scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
```

```
F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

In [74]:
```
print("The shape of training is",teacher_number_of_previously_posted_projects_train.shape, y_train.shape)
print("The shape of cv is",teacher_number_of_previously_posted_projects_cv.shape, y_cv.shape)
print("The shape of test is",teacher_number_of_previously_posted_projects_test.shape, y_test.shape)
```

```
The shape of training is (49041, 1) (49041,)
The shape of cv is (24155, 1) (24155,)
The shape of test is (36052, 1) (36052,)
```

In [75]:
```
X_train.head()
```

Out[75]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
0	107393	p041180	e0fe58a4c2c324b6c91473e846e046b7	Ms.	CA	2 0 0

1	132224	p214558	f5e105f3dbd2d2eac15f38e98daae7b8	Ms.	AZ	2 0 0
2	86372	p242163	e0740f23ac0f6a07c962682c1160e729	Ms.	IL	2 0 0
3	127080	p099474	aed8a84d9fb910a17d14e8359a2df2ed	Mrs.	CA	2 1 0
4	93419	p218892	165dd1ccb69a93b0302a8de2ee12c2de	Ms.	LA	2 0 2

Merging all the above features

SET 1 :

In [76]:
```
from scipy.sparse import hstack
X_tr = hstack((text_bow_train,price_train,title_bow_train,school_state_categ
ories_one_hot_train,pgc_one_hot_train,teacher_prefix_categories_one_hot_trai
n,categories_one_hot_train,sub_categories_one_hot_train,teacher_number_of_pr
eviously_posted_projects_train)).tocsr()
X_cr = hstack((text_bow_cv,price_cv,title_bow_cv,school_state_categories_one
_hot_cv,pgc_one_hot_cv,teacher_prefix_categories_one_hot_cv,categories_one_h
ot_cv,sub_categories_one_hot_cv,teacher_number_of_previously_posted_projects
_cv)).tocsr()
X_te = hstack((text_bow_test,price_test,title_bow_test,school_state_categori
es_one_hot_test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test,cate
gories_one_hot_test,sub_categories_one_hot_test,teacher_number_of_previously
_posted_projects_test)).tocsr()

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
(49041, 14265) (49041,)
(24155, 14265) (24155,)
(36052, 14265) (36052,)
```

SET 2 :

In [81]:
```
X_tr2 = hstack((title_tfidf_train,price_train,text_tfidf_train,school_state_
categories_one_hot_train,pgc_one_hot_train,teacher_prefix_categories_one_hot
_train,categories_one_hot_train,sub_categories_one_hot_train,teacher_number_
of_previously_posted_projects_train)).tocsr()
X_cr2 = hstack((title_tfidf_cv,price_cv,text_tfidf_cv,school_state_categorie
s_one_hot_cv,pgc_one_hot_cv,teacher_prefix_categories_one_hot_cv,categories_
one_hot_cv,sub_categories_one_hot_cv,teacher_number_of_previously_posted_pro
jects_cv)).tocsr()
X_te2 = hstack((title_tfidf_test,price_test,text_tfidf_test,school_state_cat
egories_one_hot_test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test
,categories_one_hot_test,sub_categories_one_hot_test,teacher_number_of_previ
ously_posted_projects_test)).tocsr()

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
(49041, 14265) (49041,)
(24155, 14265) (24155,)
(36052, 14265) (36052,)
```

SET 3 :

In [77]:
```
X_tr3 = hstack((avg_w2v_vectors_essay_train,price_train,avg_w2v_vectors_titl
e_train,school_state_categories_one_hot_train,pgc_one_hot_train,teacher_pref
ix_categories_one_hot_train,categories_one_hot_train,sub_categories_one_hot_
train,teacher_number_of_previously_posted_projects_train)).tocsr()
X_cr3 = hstack((avg_w2v_vectors_essay_cv,price_cv,avg_w2v_vectors_title_cv,s
chool_state_categories_one_hot_cv,pgc_one_hot_cv,teacher_prefix_categories_o
ne_hot_cv,categories_one_hot_cv,sub_categories_one_hot_cv,teacher_number_of_
previously_posted_projects_cv)).tocsr()
X_te3 = hstack((avg_w2v_vectors_essay_test,price_test,avg_w2v_vectors_title_
test,school_state_categories_one_hot_test,pgc_one_hot_test,teacher_prefix_ca
tegories_one_hot_test,categories_one_hot_test,sub_categories_one_hot_test,te
acher_number_of_previously_posted_projects_test)).tocsr()

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
(49041, 14246) (49041,)
(24155, 14246) (24155,)
(36052, 14246) (36052,)
```

SET 4 :

In [78]:
```
X_tr4 = hstack((tfidf_w2v_vectors_essay_train,price_train,tfidf_w2v_vectors_
title_train,school_state_categories_one_hot_train,pgc_one_hot_train,teacher_
prefix_categories_one_hot_train,categories_one_hot_train,sub_categories_one_
hot_train,teacher_number_of_previously_posted_projects_train)).tocsr()
X_cr4 = hstack((tfidf_w2v_vectors_essay_cv,price_cv,tfidf_w2v_vectors_title_
cv,school_state_categories_one_hot_cv,pgc_one_hot_cv,teacher_prefix_categori
es_one_hot_cv,categories_one_hot_cv,sub_categories_one_hot_cv,teacher_number
_of_previously_posted_projects_cv)).tocsr()
```

```
X_te4 = hstack((tfidf_w2v_vectors_essay_test,price_test,tfidf_w2v_vectors_title_test,school_state_categories_one_hot_test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test,categories_one_hot_test,sub_categories_one_hot_test,teacher_number_of_previously_posted_projects_test)).tocsr()

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
(49041, 14246) (49041,)
(24155, 14246) (24155,)
(36052, 14246) (36052,)
```

Assignment 3: Apply KNN

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

```
                    X_new = SelectKBest(chi2, k=20).fit_tran
sform(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

Note: Data Leakage

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

2.4.1 Applying KNN brute force on BOW, SET 1

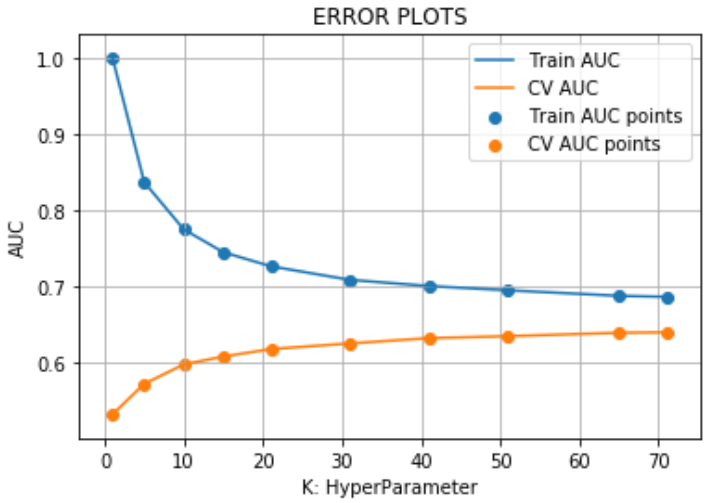
In [69]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
estimates of the positive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49
041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

Hyper Parameter Tuning

In [80]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71]
```

```
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr, y_train)

    y_train_pred = batch_predict(neigh, X_tr)
    y_cv_pred = batch_predict(neigh, X_cr)
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

```
100%|██████████| 10/10 [59:42<00:00, 354.93s/it]
```



In [81]:
```
### From the above plot the best K is :
best_k = 70
```

Training Model Using The Best Parameter :

In [82]:
```
from sklearn.metrics import roc_curve, auc


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

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

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

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

Confusion Matrix :

In [70]:
```
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(fpr*(1-tpr))]
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for thresho
ld", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [85]:
```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
test_fpr)))
```

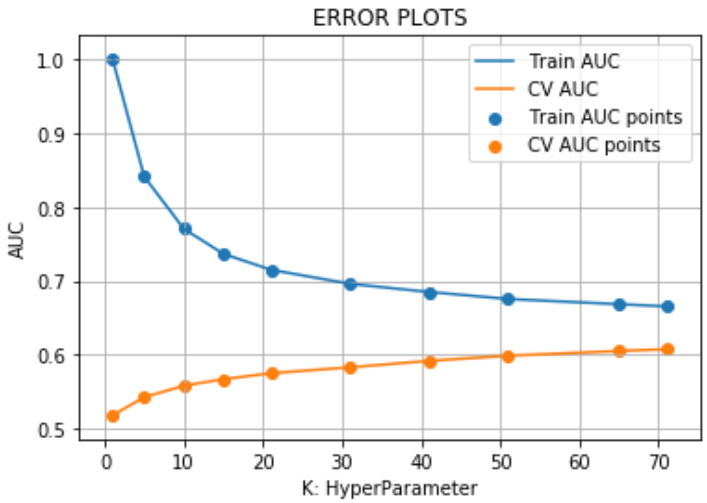
```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999591988289302 for threshold 0.771
[[ 3698  3728]
 [10173 31442]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24990395353716913 for threshold 0.786
[[ 2783  2676]
 [ 9675 20918]]
```

2.4.2 Applying KNN brute force on TFIDF, SET 2

Hyper Parameter Training :

In [86]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr2, y_train)

    y_train_pred = batch_predict(neigh, X_tr2)
    y_cv_pred = batch_predict(neigh, X_cr2)
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

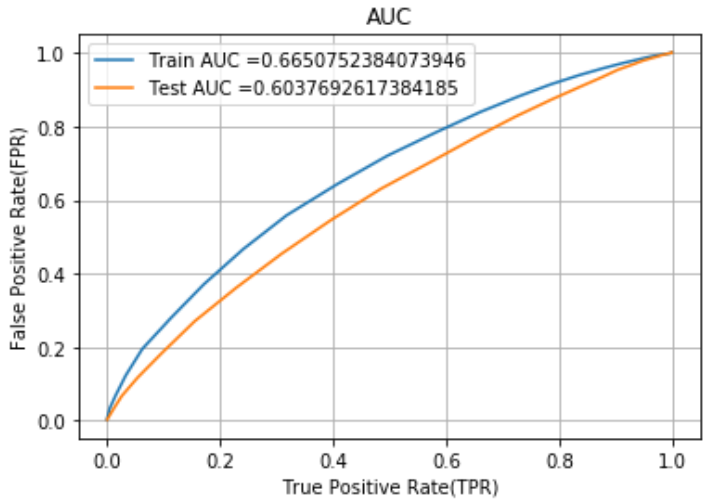
```
100%|██████████| 10/10 [58:54<00:00, 355.44s/it]
```



In [87]:
```
best_k = 72
```

In [88]:
```
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_tr2, y_train)
```

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

y_train_pred = batch_predict(neigh, X_tr2)
y_test_pred = batch_predict(neigh, X_te2)

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 [89]:
```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
test_fpr)))
```

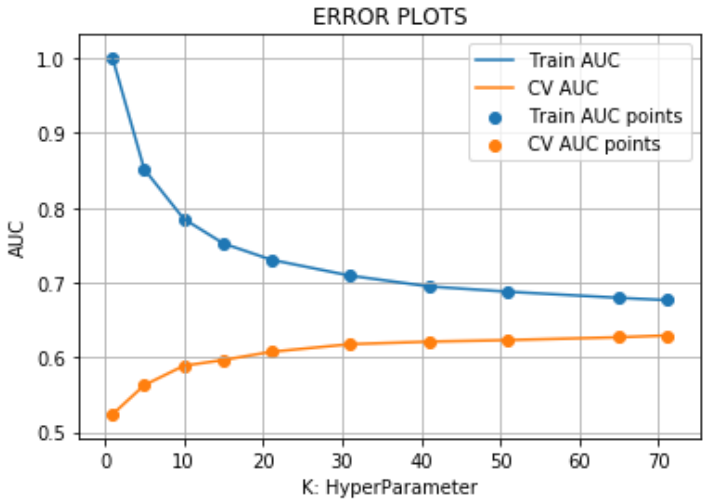
```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24998578305861388 for threshold 0.833
[[ 3741  3685]
 [11638 29977]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2497660372257936 for threshold 0.847
[[ 2813  2646]
 [11306 19287]]
```

2.4.3 Applying KNN brute force on AVG W2V, SET 3

In [90]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
```

```
train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr3, y_train)

    y_train_pred = batch_predict(neigh, X_tr3)
    y_cv_pred = batch_predict(neigh, X_cr3)
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

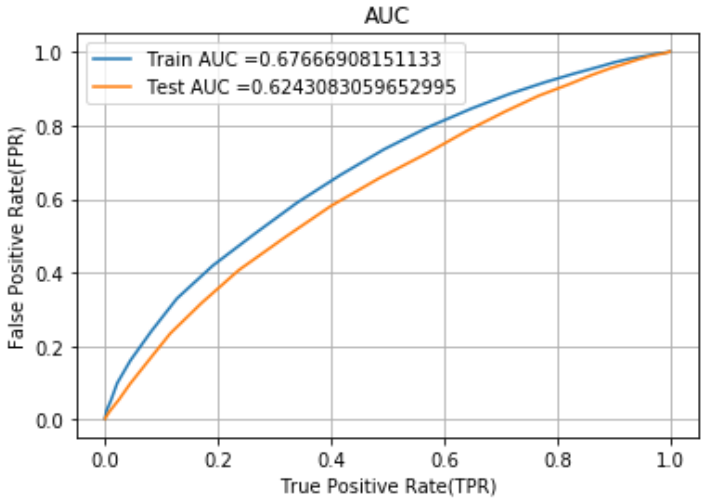
```
100%|██████████| 10/10 [6:50:03<00:00, 2461.80s/it]
```



In [91]:
```
best_k = 72
```

In [92]:
```
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_tr3, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr3)
y_test_pred = batch_predict(neigh, X_te3)

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]:
```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
test_fpr)))
```

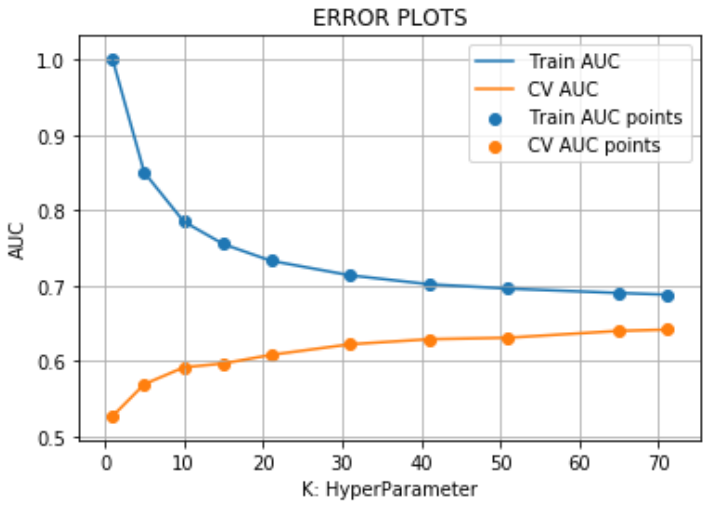
```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499632789460372 for threshold 0.833
[[ 3758  3668]
 [11024 30591]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2498137542561527 for threshold 0.847
[[ 2804  2655]
 [10478 20115]]
```

2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [80]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
train_auc = []
cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 65, 71]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr4, y_train)

    y_train_pred = batch_predict(neigh, X_tr4)
    y_cv_pred = batch_predict(neigh, X_cr4)
    train_auc.append(roc_auc_score(y_train,y_train_pred))
```

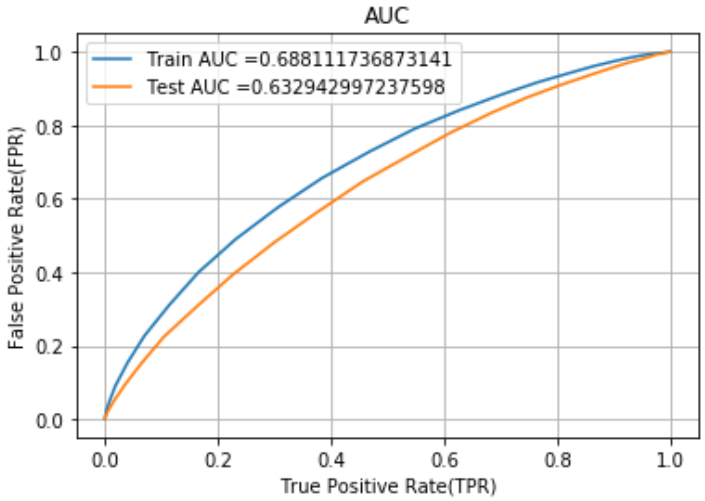
```
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

```
100%|██████████| 10/10 [6:48:17<00:00, 2449.42s/it]
```



In [81]:
```
best_k = 72
```

In [82]:
```
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_tr4, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr4)
y_test_pred = batch_predict(neigh, X_te4)

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 [85]:
```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
test_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24903235942690932 for threshold 0.833
[[ 3944  3482]
 [11277 30338]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24855519241322035 for threshold 0.833
[[ 2522  2937]
 [ 8641 21952]]
```

2.5 Feature selection with `SelectKBest`

In [71]:
```
from sklearn.preprocessing import Normalizer

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

price_train = normalizer.transform(X_train['price'].values.reshape(-1,1))
price_cv = normalizer.transform(X_cv['price'].values.reshape(-1,1))
price_test = normalizer.transform(X_test['price'].values.reshape(-1,1))
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
```

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [72]:
```
normalizer = Normalizer()
normalizer.fit(X_train['quantity'].values.reshape(-1,1))
quantity_train = normalizer.transform(X_train['quantity'].values.reshape(-1,
1))
quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(-1,1
))
```

```
print("After vectorization")
print(quantity_train.shape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
```

```
After vectorization
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [73]:
```
normalizer = Normalizer()
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].value
s.reshape(-1,1))
prev_projects_train = normalizer.transform(X_train['teacher_number_of_previo
usly_posted_projects'].values.reshape(-1,1))
prev_projects_cv = normalizer.transform(X_cv['teacher_number_of_previously_p
osted_projects'].values.reshape(-1,1))
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previous
ly_posted_projects'].values.reshape(-1,1))
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [74]:
```
from scipy.sparse import hstack
X_tr2 = hstack((title_tfidf_train,price_train,text_tfidf_train,school_state_
categories_one_hot_train,pgc_one_hot_train,teacher_prefix_categories_one_hot
_train,categories_one_hot_train,sub_categories_one_hot_train,prev_projects_t
rain)).tocsr()
X_cr2 = hstack((title_tfidf_cv,price_cv,text_tfidf_cv,school_state_categorie
s_one_hot_cv,pgc_one_hot_cv,teacher_prefix_categories_one_hot_cv,categories_
one_hot_cv,sub_categories_one_hot_cv,prev_projects_cv)).tocsr()
X_te2 = hstack((title_tfidf_test,price_test,text_tfidf_test,school_state_cat
egories_one_hot_test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test
,categories_one_hot_test,sub_categories_one_hot_test,prev_projects_test)).to
csr()

print("Final Data matrix")
print(X_tr2.shape, y_train.shape)
print(X_cr2.shape, y_cv.shape)
print(X_te2.shape, y_test.shape)
```

```
Final Data matrix
(49041, 14187) (49041,)
(24155, 14187) (24155,)
(36052, 14187) (36052,)
```

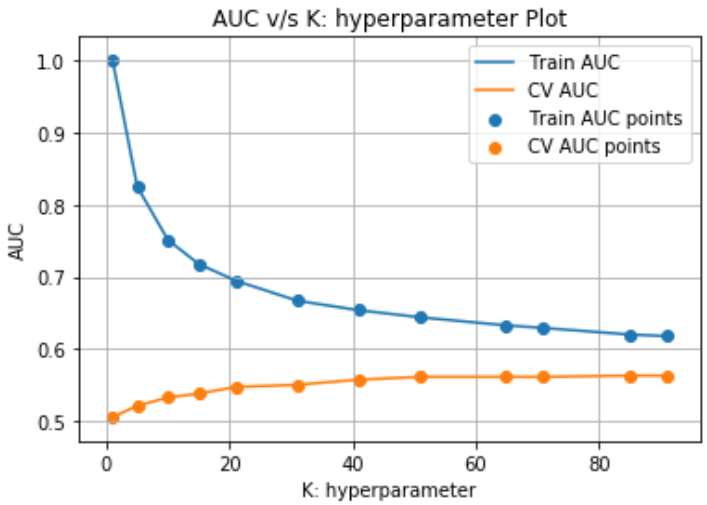
In [76]:
```
from sklearn.feature_selection import SelectKBest, chi2

X_tr_n=SelectKBest(chi2, k=2000).fit(X_tr2, y_train)
X_tr_new=X_tr_n.transform(X_tr2)
X_te_new=X_tr_n.transform(X_te2)
X_cr_new=X_tr_n.transform(X_cr2)
print("The shape of the respective matrix are:",X_tr_new.shape)
print("The shape of the respective matrix are:",X_te_new.shape)
print("The shape of the respective matrix are:",X_cr_new.shape)
```

```
The shape of the respective matrix are: (49041, 2000)
The shape of the respective matrix are: (36052, 2000)
```

```
The shape of the respective matrix are: (24155, 2000)
```

In [77]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
train_auc = []
cv_auc = []

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

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

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

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

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

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

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

```
100%|██████████| 12/12 [36:33<00:00, 181.37s/it]
```



In [78]:
```
best_k = 85
```

In [79]:
```
neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(X_tr_new, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
```

```
# not the predicted outputs

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

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

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

In [80]:
```
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr)))
```

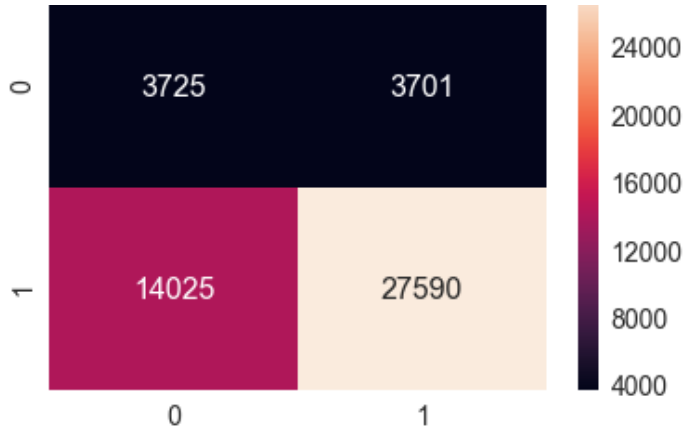
```
the maximum value of tpr*(1-fpr) 0.24999738872505156 for threshold 0.835
[[ 3725  3701]
 [14025 27590]]
```

In [81]:
```
conf_matr_df_ = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred,
tr_thresholds, train_fpr, train_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24999738872505156 for threshold 0.835
```

In [82]:
```
sns.set(font_scale=1.4)
sns.heatmap(conf_matr_df_, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[82]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1831364c9e8>
```



In [83]:
```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

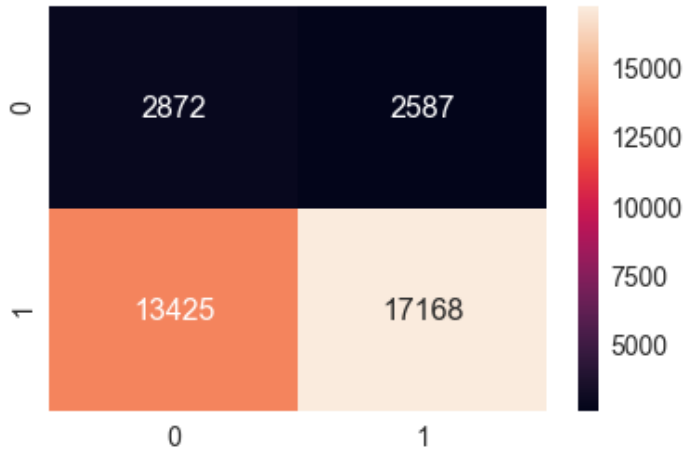
```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24931859778640628 for threshold 0.847
[[ 2872  2587]
 [13425 17168]]
```

In [84]:
```
conf_matr_df = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)), range(2),range(2))
```

```
the maximum value of tpr*(1-fpr) 0.24931859778640628 for threshold 0.847
```

In [85]:
```
sns.set(font_scale=1.4)
sns.heatmap(conf_matr_df, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[85]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1831390d160>
```



3. Conclusions

In [86]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]

x.add_row(["BOW(USING STANDARD SCALER)", "Brute", 70, 0.64])
x.add_row(["TFIDF(USING STANDARD SCALER)", "Brute", 72, 0.62])
x.add_row(["AVG W2V(USING STANDARD SCALER)", "Brute", 72, 0.62])
x.add_row(["TFIDF W2V(USING STANDARD SCALER)", "Brute", 72, 0.63])
x.add_row(["TFIDF(USING NORMALISATION)", "Top 2000", 85, 0.56])
```

```
print(x)
```

```
+-----------------------------------+----------+-----------------+------+
|            Vectorizer             |  Model   | Hyper Parameter | AUC  |
+-----------------------------------+----------+-----------------+------+
|    BOW(USING STANDARD SCALER)     |  Brute   |        70       | 0.64 |
|   TFIDF(USING STANDARD SCALER)    |  Brute   |        72       | 0.62 |
|  AVG W2V(USING STANDARD SCALER)   |  Brute   |        72       | 0.62 |
| TFIDF W2V(USING STANDARD SCALER)  |  Brute   |        72       | 0.63 |
|    TFIDF(USING NORMALISATION)     | Top 2000 |        85       | 0.56 |
+-----------------------------------+----------+-----------------+------+
```