# 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 | De |
| --- | --- |
| **project_id** | A unique identifier for the proposed project. **Example:** |
| **project_title** | Title of the project. **E**<br>• `Art Will Make You`<br>• `First Gr` |
| **project_grade_category** | Grade level of students for which the project is targeted. One of the enumerate<br>• `Grades`<br>• `Gra`<br>• `Gra`<br>• `Grad` |
| **project_subject_categories** | One or more (comma-separated) subject categories for the projec following enumerated list<br>• `Applied L`<br>• `Care &`<br>• `Health &`<br>• `History &`<br>• `Literacy & L`<br>• `Math &`<br>• `Music & T`<br>• `Specia`<br>• <br><br>**Ex**<br>• `Music & T`<br>• `Literacy & Language, Math &` |
| **school_state** | State where school is located (Two-letter U.S. p (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Posta **Exar** |
| **project_subject_subcategories** | One or more (comma-separated) subject subcategories for th **E**<br>• `L`<br>• `Literature & Writing, Social S` |
| **project_resource_summary** | An explanation of the resources needed for the project. **E**<br>• `My students need hands on literacy materials to sensory` |
| **project_essay_1** | First applicat |
| **project_essay_2** | Second applicat |
| **project_essay_3** | Third applicat |
| **project_essay_4** | Fourth applicat |
| **project_submitted_datetime** | Datetime when project application was submitted. **Example:** `201 12:43` |
| **teacher_id** | A unique identifier for the teacher of the proposed project. `bdf8baa8fedef6bfeec7ae4ff` |

| Feature | De |
|---|---|
| `teacher_prefix` | Teacher's title. One of the following enumerate<br>• <br>• <br>• <br>• <br>• <br>• T |
| `teacher_number_of_previously_posted_projects` | Number of project applications previously submitted by the sam<br>**Exa** |

* 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 |

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

## 1.1 Reading Data

In [2]:

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

In [3]:

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

```
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/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.col
umns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
39
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_state |
|---|---|---|---|---|---|
| **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/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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 ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        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://stackoverflow.com/a/47
301924/4084039

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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 ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

## 1.3 Text preprocessing

In [8]:

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

In [9]:

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

Out[9]:

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

In [10]:

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

In [11]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

I have been fortunate enough to use the Fairy Tale STEM kits in my classro
om as well as the STEM journals, which my students really enjoyed.  I woul
d love to implement more of the Lakeshore STEM kits in my classroom for th
e next school year as they provide excellent and engaging STEM lessons.My
students come from a variety of backgrounds, including language and socioe
conomic status.  Many of them don't have a lot of experience in science an
d engineering and these kits give me the materials to provide these exciti
ng opportunities for my students.Each month I try to do several science or
STEM/STEAM projects.  I would use the kits and robot to help guide my scie
nce instruction in engaging and meaningful ways.  I can adapt the kits to
my current language arts pacing guide where we already teach some of the m
aterial in the kits like tall tales (Paul Bunyan) or Johnny Appleseed.  Th
e following units will be taught in the next school year where I will impl
ement these kits: magnets, motion, sink vs. float, robots.  I often get to
these units and don't know If I am teaching the right way or using the rig
ht materials.    The kits will give me additional ideas, strategies, and l
essons to prepare my students in science.It is challenging to develop high
quality science activities.  These kits give me the materials I need to pr
ovide my students with science activities that will go along with the curr
iculum in my classroom.  Although I have some things (like magnets) in my
classroom, I don't know how to use them effectively.  The kits will provid
e me with the right amount of materials and show me how to use them in an
appropriate way.
==========
I teach high school English to students with learning and behavioral disab
ilities. My students all vary in their ability level. However, the ultimat
e goal is to increase all students literacy levels. This includes their re
ading, writing, and communication levels.I teach a really dynamic group of
students. However, my students face a lot of challenges. My students all l
ive in poverty and in a dangerous neighborhood. Despite these challenges,
I have students who have the the desire to defeat these challenges. My stu
dents all have learning disabilities and currently all are performing belo
w grade level. My students are visual learners and will benefit from a cla
ssroom that fulfills their preferred learning style.The materials I am req
uesting will allow my students to be prepared for the classroom with the n
ecessary supplies.  Too often I am challenged with students who come to sc
hool unprepared for class due to economic challenges.  I want my students
to be able to focus on learning and not how they will be able to get schoo
l supplies.  The supplies will last all year.  Students will be able to co
mplete written assignments and maintain a classroom journal.  The chart pa
per will be used to make learning more visual in class and to create poste
rs to aid students in their learning.  The students have access to a class
room printer.  The toner will be used to print student work that is comple
ted on the classroom Chromebooks.I want to try and remove all barriers for
the students learning and create opportunities for learning. One of the bi
ggest barriers is the students not having the resources to get pens, pape
r, and folders. My students will be able to increase their literacy skills
because of this project.
==========
\"Life moves pretty fast. If you don't stop and look around once in awhil
e, you could miss it.\"  from the movie, Ferris Bueller's Day Off.  Think
back...what do you remember about your grandparents?  How amazing would it
be to be able to flip through a book to see a day in their lives?My second
graders are voracious readers! They love to read both fiction and nonficti
on books.  Their favorite characters include Pete the Cat, Fly Guy, Piggie
and Elephant, and Mercy Watson. They also love to read about insects, spac
e and plants. My students are hungry bookworms! My students are eager to l
earn and read about the world around them. My kids love to be at school an
d are like little sponges absorbing everything around them. Their parents
work long hours and usually do not see their children. My students are usu
ally cared for by their grandparents or a family friend. Most of my studen

ts do not have someone who speaks English at home. Thus it is difficult fo
r my students to acquire language.Now think forward... wouldn't it mean a
lot to your kids, nieces or nephews or grandchildren, to be able to see a
day in your life today 30 years from now? Memories are so precious to us a
nd being able to share these memories with future generations will be a re
warding experience.  As part of our social studies curriculum, students wi
ll be learning about changes over time.  Students will be studying photos
to learn about how their community has changed over time.  In particular,
we will look at photos to study how the land, buildings, clothing, and sch
ools have changed over time.  As a culminating activity, my students will
capture a slice of their history and preserve it through scrap booking. Ke
y important events in their young lives will be documented with the date,
location, and names.  Students will be using photos from home and from sc
hool to create their second grade memories.  Their scrap books will prese
rve their unique stories for future generations to enjoy.Your donation to
this project will provide my second graders with an opportunity to learn a
bout social studies in a fun and creative manner.  Through their scrapbook
s, children will share their story with others and have a historical docum
ent for the rest of their lives.
==========
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the small
est students with the biggest enthusiasm for learning. My students learn i
n many different ways using all of our senses and multiple intelligences.
I use a wide range of techniques to help all my students succeed. \r\nStud
ents in my class come from a variety of different backgrounds which makes
for wonderful sharing of experiences and cultures, including Native Americ
ans.\r\nOur school is a caring community of successful learners which can
be seen through collaborative student project based learning in and out of
the classroom. Kindergarteners in my class love to work with hands-on mate
rials and have many different opportunities to practice a skill before it
is mastered. Having the social skills to work cooperatively with friends i
s a crucial aspect of the kindergarten curriculum.Montana is the perfect p
lace to learn about agriculture and nutrition. My students love to role pl
ay in our pretend kitchen in the early childhood classroom. I have had sev
eral kids ask me, \"Can we try cooking with REAL food?\" I will take their
idea and create \"Common Core Cooking Lessons\" where we learn important m
ath and writing concepts while cooking delicious healthy food for snack ti
me. My students will have a grounded appreciation for the work that went i
nto making the food and knowledge of where the ingredients came from as we
ll as how it's healthy for their bodies. This project would expand our lea
rning of nutrition and agricultural cooking recipes by having us peel our
own apples to make homemade applesauce, make our own bread, and mix up hea
lthy plants from our classroom garden in the spring. We will also create o
ur own cookbooks to be printed and shared with families. \r\nStudents will
gain math and literature skills as well as a life long enjoyment for healt
hy cooking.nannan
==========
My classroom consists of twenty-two amazing sixth graders from different c
ultures and backgrounds. They are a social bunch who enjoy working in part
ners and working with groups. They are hard-working and eager to head to m
iddle school next year. My job is to get them ready to make this transitio
n and make it as smooth as possible. In order to do this, my students need
to come to school every day and feel safe and ready to learn. Because they
are getting ready to head to middle school, I give them lots of choice- ch
oice on where to sit and work, the order to complete assignments, choice o
f projects, etc. Part of the students feeling safe is the ability for them
to come into a welcoming, encouraging environment. My room is colorful and
the atmosphere is casual. I want them to take ownership of the classroom b
ecause we ALL share it together. Because my time with them is limited, I w
ant to ensure they get the most of this time and enjoy it to the best of t
heir abilities.Currently, we have twenty-two desks of differing sizes, yet

```
the desks are similar to the ones the students will use in middle school.
We also have a kidney table with crates for seating. I allow my students t
o choose their own spots while they are working independently or in group
s. More often than not, most of them move out of their desks and onto the
crates. Believe it or not, this has proven to be more successful than maki
ng them stay at their desks! It is because of this that I am looking towar
d the “Flexible Seating” option for my classroom.\r\n The students look fo
rward to their work time so they can move around the room. I would like to
get rid of the constricting desks and move toward more “fun” seating optio
ns. I am requesting various seating so my students have more options to si
t. Currently, I have a stool and a papasan chair I inherited from the prev
ious sixth-grade teacher as well as five milk crate seats I made, but I wo
uld like to give them more options and reduce the competition for the “goo
d seats”. I am also requesting two rugs as not only more seating options b
ut to make the classroom more welcoming and appealing. In order for my stu
dents to be able to write and complete work without desks, I am requesting
a class set of clipboards. Finally, due to curriculum that requires groups
to work together, I am requesting tables that we can fold up when we are n
ot using them to leave more room for our flexible seating options.\r\nI kn
ow that with more seating options, they will be that much more excited abo
ut coming to school! Thank you for your support in making my classroom one
students will remember forever!nannan
==================================================
```

In [12]:

```
# 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 [13]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smal
lest students with the biggest enthusiasm for learning. My students learn
in many different ways using all of our senses and multiple intelligences.
I use a wide range of techniques to help all my students succeed. \r\nStud
ents in my class come from a variety of different backgrounds which makes
for wonderful sharing of experiences and cultures, including Native Americ
ans.\r\nOur school is a caring community of successful learners which can
be seen through collaborative student project based learning in and out of
the classroom. Kindergarteners in my class love to work with hands-on mate
rials and have many different opportunities to practice a skill before it
is mastered. Having the social skills to work cooperatively with friends i
s a crucial aspect of the kindergarten curriculum.Montana is the perfect p
lace to learn about agriculture and nutrition. My students love to role pl
ay in our pretend kitchen in the early childhood classroom. I have had sev
eral kids ask me, \"Can we try cooking with REAL food?\" I will take their
idea and create \"Common Core Cooking Lessons\" where we learn important m
ath and writing concepts while cooking delicious healthy food for snack ti
me. My students will have a grounded appreciation for the work that went i
nto making the food and knowledge of where the ingredients came from as we
ll as how it is healthy for their bodies. This project would expand our le
arning of nutrition and agricultural cooking recipes by having us peel our
own apples to make homemade applesauce, make our own bread, and mix up hea
lthy plants from our classroom garden in the spring. We will also create o
ur own cookbooks to be printed and shared with families. \r\nStudents will
gain math and literature skills as well as a life long enjoyment for healt
hy cooking.nannan
==================================================
```

In [14]:

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

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smalle
st students with the biggest enthusiasm for learning. My students learn in
many different ways using all of our senses and multiple intelligences. I
use a wide range of techniques to help all my students succeed.  Students
in my class come from a variety of different backgrounds which makes for w
onderful sharing of experiences and cultures, including Native Americans.
Our school is a caring community of successful learners which can be seen
through collaborative student project based learning in and out of the cla
ssroom. Kindergarteners in my class love to work with hands-on materials a
nd have many different opportunities to practice a skill before it is mast
ered. Having the social skills to work cooperatively with friends is a cru
cial aspect of the kindergarten curriculum.Montana is the perfect place to
learn about agriculture and nutrition. My students love to role play in ou
r pretend kitchen in the early childhood classroom. I have had several kid
s ask me,  Can we try cooking with REAL food?  I will take their idea and
create  Common Core Cooking Lessons  where we learn important math and wri
ting concepts while cooking delicious healthy food for snack time. My stud
ents will have a grounded appreciation for the work that went into making
the food and knowledge of where the ingredients came from as well as how i
t is healthy for their bodies. This project would expand our learning of n
utrition and agricultural cooking recipes by having us peel our own apples
to make homemade applesauce, make our own bread, and mix up healthy plants
from our classroom garden in the spring. We will also create our own cookb
ooks to be printed and shared with families.  Students will gain math and
literature skills as well as a life long enjoyment for healthy cooking.nan
nan
```

In [15]:

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

```
 A person is a person no matter how small Dr Seuss I teach the smallest st
udents with the biggest enthusiasm for learning My students learn in many
different ways using all of our senses and multiple intelligences I use a
wide range of techniques to help all my students succeed Students in my cl
ass come from a variety of different backgrounds which makes for wonderful
sharing of experiences and cultures including Native Americans Our school
is a caring community of successful learners which can be seen through col
laborative student project based learning in and out of the classroom Kind
ergarteners in my class love to work with hands on materials and have many
different opportunities to practice a skill before it is mastered Having t
he social skills to work cooperatively with friends is a crucial aspect of
the kindergarten curriculum Montana is the perfect place to learn about ag
riculture and nutrition My students love to role play in our pretend kitch
en in the early childhood classroom I have had several kids ask me Can we
try cooking with REAL food I will take their idea and create Common Core C
ooking Lessons where we learn important math and writing concepts while co
oking delicious healthy food for snack time My students will have a ground
ed appreciation for the work that went into making the food and knowledge
of where the ingredients came from as well as how it is healthy for their
bodies This project would expand our learning of nutrition and agricultura
l cooking recipes by having us peel our own apples to make homemade apples
auce make our own bread and mix up healthy plants from our classroom garde
n in the spring We will also create our own cookbooks to be printed and sh
ared with families Students will gain math and literature skills as well a
s a life long enjoyment for healthy cooking nannan
```

In [16]:

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

In [17]:

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

```
100%|██████████| 109248/109248 [02:21<00:00, 771.73it/s]
```

In [18]:

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

Out[18]:

```
'person person no matter small dr seuss teach smallest students biggest en
thusiasm learning students learn many different ways using senses multiple
intelligences use wide range techniques help students succeed students cla
ss come variety different backgrounds makes wonderful sharing experiences
cultures including native americans school caring community successful lea
rners seen collaborative student project based learning classroom kinderga
rteners class love work hands materials many different opportunities pract
ice skill mastered social skills work cooperatively friends crucial aspect
kindergarten curriculum montana perfect place learn agriculture nutrition
students love role play pretend kitchen early childhood classroom several
kids ask try cooking real food take idea create common core cooking lesson
s learn important math writing concepts cooking delicious healthy food sna
ck time students grounded appreciation work went making food knowledge ing
redients came well healthy bodies project would expand learning nutrition
agricultural cooking recipes us peel apples make homemade applesauce make
bread mix healthy plants classroom garden spring also create cookbooks pri
nted shared families students gain math literature skills well life long e
njoyment healthy cooking nannan'
```

## 1.4 Preprocessing of `project_title`

In [19]:

```
# similarly you can preprocess the titles also
```

In [20]:

```
pt = project_data['project_title']
```

In [21]:

```
pt.head()
```

Out[21]:

```
55660      Engineering STEAM into the Primary Classroom
76127                           Sensory Tools for Focus
51140    Mobile Learning with a Mobile Listening Center
473              Flexible Seating for Flexible Learning
41558            Going Deep: The Art of Inner Thinking!
Name: project_title, dtype: object
```

In [22]:

```
pt.nunique()
```

Out[22]:

```
100851
```

In [23]:

```
pt.values[100]
```

Out[23]:

```
'iCan with iPads...and YOU!'
```

In [24]:

```
# 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 [25]:

```
sent = decontracted(pt.values[2000])
print(sent)
print("="*50)
```

```
Empowering Students through Art in the Makerspace
==================================================
```

In [26]:

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

```
Empowering Students through Art in the Makerspace
```

In [27]:

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

```
Empowering Students through Art in the Makerspace
```

In [28]:

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

In [29]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(pt.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 not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:06<00:00, 18176.18it/s]
```

In [30]:

```
preprocessed_titles[2000:2010]
```

Out[30]:

```
['empowering students art makerspace',
 'tablet astic',
 'election fall 2016 materials',
 'whole brain learning lounge',
 'calculators help us fractions algebra geometry more',
 'just basics',
 'alternative seating guru need rugs thanksteach',
 'capture experiences',
 'breakout ordinary',
 '21st century classroom makeover']
```

In [ ]:

```

```

## 1.5 Preparing data for models

In [31]:

```
project_data.columns
```

Out[31]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       '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

In [32]:

```
project_data.columns
```

Out[32]:

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

### 1.5.1 Vectorizing Categorical data

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

In [33]:

```
'''
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
 binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
'''
```

Out[33]:

```
'\n# we use count vectorizer to convert the values into one \nfrom sklear
n.feature_extraction.text import CountVectorizer\nvectorizer = CountVector
izer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=Tru
e)\ncategories_one_hot = vectorizer.fit_transform(project_data[\'clean_cat
egories\'].values)\nprint(vectorizer.get_feature_names())\nprint("Shape of
matrix after one hot encodig ",categories_one_hot.shape)\n'
```

In [34]:

```
'''
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal
se, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].v
alues)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)

'''
```

Out[34]:

```
'\n# we use count vectorizer to convert the values into one \nvectorizer =
CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal
se, binary=True)\nsub_categories_one_hot = vectorizer.fit_transform(projec
t_data[\'clean_subcategories\'].values)\nprint(vectorizer.get_feature_name
s())\nprint("Shape of matrix after one hot encodig ",sub_categories_one_ho
t.shape)\n\n'
```

In [35]:

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

## school state

In [36]:

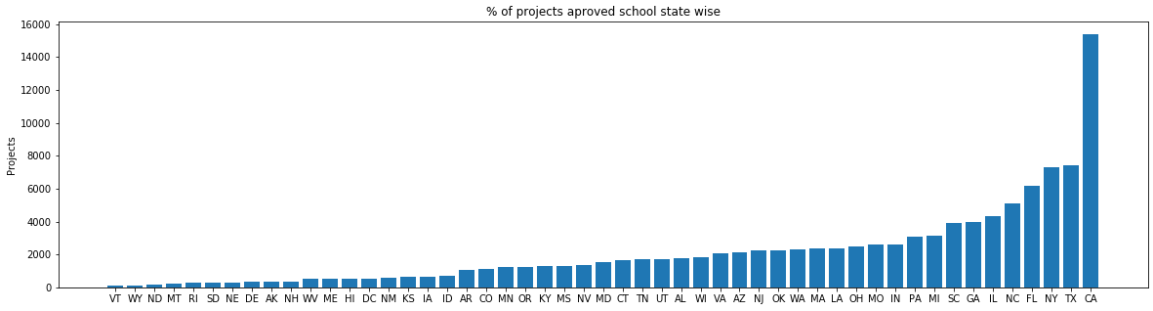
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())
```

In [37]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_scl_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))


ind = np.arange(len(sorted_scl_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(sorted_scl_dict.values()))

plt.ylabel('Projects')
plt.title('% of projects aproved school state wise')
plt.xticks(ind, list(sorted_scl_dict.keys()))
plt.show()
```




In [38]:

```
'''
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_scl_dict.keys()), lowercase=False,
 binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot_1 = vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_1.shape)

'''
```

Out[38]:

```
'\n# we use count vectorizer to convert the values into one hot encoded fe
atures\nvectorizer = CountVectorizer(vocabulary=list(sorted_scl_dict.keys
()), lowercase=False, binary=True)\nvectorizer.fit(project_data[\'school_s
tate\'].values)\nprint(vectorizer.get_feature_names())\n\n\nsub_categories
_one_hot_1 = vectorizer.transform(project_data[\'school_state\'].values)\n
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_1.sh
ape)\n\n'
```

## teacher prefix

In [39]:

```
project_data.groupby(['teacher_prefix'])['teacher_prefix'].count()
```

Out[39]:

```
teacher_prefix
Dr.           13
Mr.        10648
Mrs.       57269
Ms.        38955
Teacher     2360
Name: teacher_prefix, dtype: int64
```

In [40]:

```
project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

Out[40]:

```
30368    NaN
57654    NaN
7820     NaN
Name: teacher_prefix, dtype: object
```

In [41]:

```
project_data['teacher_prefix'].fillna(project_data['teacher_prefix'].mode()[0],inplace=
True)
```

In [42]:

```
project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

Out[42]:

```
Series([], Name: teacher_prefix, dtype: object)
```

In [92]:

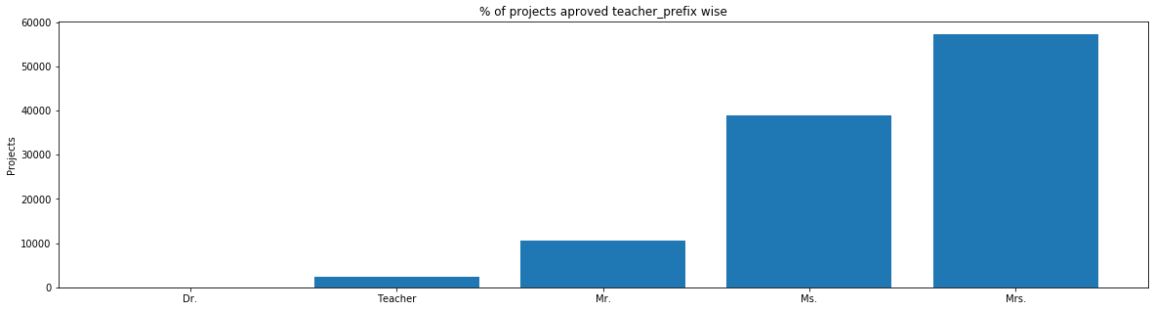
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['teacher_prefix'].values:
    my_counter.update(word.split())
```

In [93]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_tp_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))


ind = np.arange(len(sorted_tp_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(sorted_tp_dict.values()))

plt.ylabel('Projects')
plt.title('% of projects aproved teacher_prefix wise')
plt.xticks(ind, list(sorted_tp_dict.keys()))
plt.show()
```




In [45]:

```
'''
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_tp_dict.keys()), lowercase=False, b
inary=True)
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot_2 = vectorizer.transform(project_data['teacher_prefix'].values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_2.shape)

'''
```

Out[45]:

```
'\n# we use count vectorizer to convert the values into one hot encoded fe
atures\nvectorizer = CountVectorizer(vocabulary=list(sorted_tp_dict.keys
()), lowercase=False, binary=True)\nvectorizer.fit(project_data[\'teacher_
prefix\'].values)\nprint(vectorizer.get_feature_names())\n\n\nsub_categori
es_one_hot_2 = vectorizer.transform(project_data[\'teacher_prefix\'].value
s)\nprint("Shape of matrix after one hot encodig ",sub_categories_one_hot_
2.shape)\n\n'
```

## project grade category

In [46]:

```
project_data['project_grade_category'][project_data['project_grade_category'].isnull()==True]
```

Out[46]:

```
Series([], Name: project_grade_category, dtype: object)
```

In [87]:

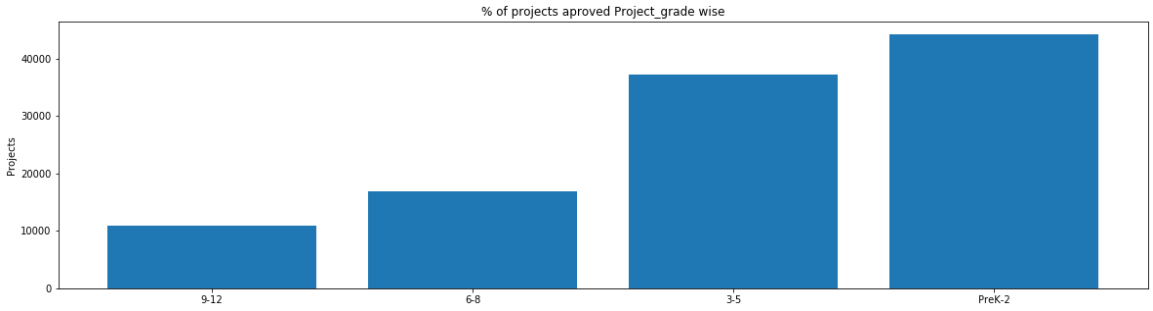
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['project_grade_category'].values:
    my_counter.update(word.split())
```

In [90]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_pgc_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
del sorted_pgc_dict["Grades"]

ind = np.arange(len(sorted_tp_dict))
plt.figure(figsize=(20,5))
p1 = plt.bar(ind, list(sorted_tp_dict.values()))

plt.ylabel('Projects')
plt.title('% of projects aproved Project_grade wise')
plt.xticks(ind, list(sorted_tp_dict.keys()))
plt.show()
```




In [ ]:

In [49]:

```
'''
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_tp_dict.keys()), lowercase=False, b
inary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot_3 = vectorizer.transform(project_data['project_grade_category'].
values)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_3.shape)

'''
```

Out[49]:

```
'\n# we use count vectorizer to convert the values into one hot encoded fe
atures\nvectorizer = CountVectorizer(vocabulary=list(sorted_tp_dict.keys
()), lowercase=False, binary=True)\nvectorizer.fit(project_data[\'project_
grade_category\'].values)\nprint(vectorizer.get_feature_names())\n\n\nsub_
categories_one_hot_3 = vectorizer.transform(project_data[\'project_grade_c
ategory\'].values)\nprint("Shape of matrix after one hot encodig ",sub_cat
egories_one_hot_3.shape)\n\n'
```

In [ ]:

```

```

In [ ]:

```

```

## 1.5.2 Vectorizing Text data

### 1.5.2.1 Bag of words

In [50]:

```
'''
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)

'''
```

Out[50]:

```
'\n# We are considering only the words which appeared in at least 10 docum
ents(rows or projects).\nvectorizer = CountVectorizer(min_df=10)\ntext_bow
= vectorizer.fit_transform(preprocessed_essays)\nprint("Shape of matrix af
ter one hot encodig ",text_bow.shape)\n\n'
```

In [51]:

```
'''
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it

vectorizer1 = CountVectorizer()
text_bow1 = vectorizer1.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",text_bow1.shape)

'''
```

Out[51]:

```
'\n# you can vectorize the title also \n# before you vectorize the title m
ake sure you preprocess it\n\nvectorizer1 = CountVectorizer()\ntext_bow1 =
vectorizer1.fit_transform(preprocessed_titles)\nprint("Shape of matrix aft
er one hot encodig ",text_bow1.shape)\n\n'
```

#### 1.5.2.2 TFIDF vectorizer

In [52]:

```
'''
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)

'''
```

Out[52]:

```
'\nfrom sklearn.feature_extraction.text import TfidfVectorizer\nvectorizer
= TfidfVectorizer(min_df=10)\ntext_tfidf = vectorizer.fit_transform(prepro
cessed_essays)\nprint("Shape of matrix after one hot encodig ",text_tfidf.
shape)\n\n'
```

In [53]:

```
'''
# Similarly you can vectorize for title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer2 = TfidfVectorizer()
text_tfidf2 = vectorizer2.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",text_tfidf2.shape)

'''
```

Out[53]:

```
'\n# Similarly you can vectorize for title also\nfrom sklearn.feature_extr
action.text import TfidfVectorizer\nvectorizer2 = TfidfVectorizer()\ntext_
tfidf2 = vectorizer2.fit_transform(preprocessed_titles)\nprint("Shape of m
atrix after one hot encodig ",text_tfidf2.shape)\n\n'
```

#### 1.5.2.3 Using Pretrained Models: Avg W2V

In [54]:

```
'''
# 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')

# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

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

for i in preproced_titles:
    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 coupus", \
      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/how-to-use-p
ickle-to-save-and-load-variables-in-python/

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


'''
```

Out[54]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Mode
l")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    f
or line in tqdm(f):\n        splitLine = line.split()\n        word = spli
tLine[0]\n        embedding = np.array([float(val) for val in splitLine
[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," w
ords loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.
txt\')\n\n# ============================\nOutput:\n    \nLoading Glove Mod
el\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ====
========================\n\nwords = []\nfor i in preproced_texts:\n    wor
ds.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend
(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords
= set(words)\nprint("the unique words in the coupus", len(words))\n\ninter
_words = set(model.keys()).intersection(words)\nprint("The number of words
that are present in both glove vectors and our coupus",       len(inter_wo
rds),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpu
s = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in word
s_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length",
len(words_courpus))\n\n\n# stronging variables into pickle files python: h
ttp://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-
python/\n\nimport pickle \nwith open(\'glove_vectors\', \'wb\') as f:\n
pickle.dump(words_courpus, f)\n\n\n'
```

In [55]:

```
'''
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-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())

'''
```

Out[55]:

```
"\n# stronging variables into pickle files python: http://www.jessicayung.
com/how-to-use-pickle-to-save-and-load-variables-in-python/\n# make sure y
ou have the glove_vectors file\nwith open('../glove_vectors', 'rb') as
f:\n    model = pickle.load(f)\n    glove_words =  set(model.keys())\n
\n"
```

In [56]:

```
'''
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # 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.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))

'''
```

Out[56]:

```
'\n# average Word2Vec\n# compute average word2vec for each review.\navg_w2
v_vectors = []; # the avg-w2v for each sentence/review is stored in this l
ist\nfor sentence in tqdm(preprocessed_essays): # for each review/sentence
\n    vector = np.zeros(300) # as word vectors are of zero length\n    cnt
_words =0; # num of words with a valid vector in the sentence/review\n
for word in sentence.split(): # for each word in a review/sentence\n
if word in glove_words:\n            vector += model[word]\n            cn
t_words += 1\n    if cnt_words != 0:\n        vector /= cnt_words\n    avg
_w2v_vectors.append(vector)\n\nprint(len(avg_w2v_vectors))\nprint(len(avg_
w2v_vectors[0]))\n\n'
```

#### 1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [57]:

```
'''
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# 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())

'''
```

Out[57]:

```
'\n# S = ["abc def pqr", "def def def abc", "pqr pqr def"]\ntfidf_model =
TfidfVectorizer()\ntfidf_model.fit(preprocessed_essays)\n# we are converti
ng a dictionary with word as a key, and the idf as a value\ndictionary = d
ict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))\ntfidf_w
ords = set(tfidf_model.get_feature_names())\n\n'
```

In [58]:

```
'''
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))

'''
```

Out[58]:

```
'\n# average Word2Vec\n# compute average word2vec for each review.\ntfidf_
w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this
list\nfor sentence in tqdm(preprocessed_essays): # for each review/sentenc
e\n    vector = np.zeros(300) # as word vectors are of zero length\n    tf
_idf_weight =0; # num of words with a valid vector in the sentence/review
\n    for word in sentence.split(): # for each word in a review/sentence\n
if (word in glove_words) and (word in tfidf_words):\n            vec = mod
el[word] # getting the vector for each word\n            # here we are mul
tiplying idf value(dictionary[word]) and the tf value((sentence.count(wor
d)/len(sentence.split())))\n            tf_idf = dictionary[word]*(sentenc
e.count(word)/len(sentence.split())) # getting the tfidf value for each wo
rd\n            vector += (vec * tf_idf) # calculating tfidf weighted w2v
\n            tf_idf_weight += tf_idf\n    if tf_idf_weight != 0:\n
vector /= tf_idf_weight\n    tfidf_w2v_vectors.append(vector)\n\nprint(len
(tfidf_w2v_vectors))\nprint(len(tfidf_w2v_vectors[0]))\n\n'
```

In [59]:

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

### 1.5.3 Vectorizing Numerical features

In [60]:

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

In [61]:

```
'''
# 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_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ...
 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1,
 1))

'''
```

Out[61]:

```
'\n# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s\n#
standardization sklearn: https://scikit-learn.org/stable/modules/generate
d/sklearn.preprocessing.StandardScaler.html\nfrom sklearn.preprocessing im
port StandardScaler\n\n# price_standardized = standardScalar.fit(project_d
ata[\'price\'].values)\n# this will rise the error\n# ValueError: Expected
2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   28
7.73   5.5 ].\n# Reshape your data either using array.reshape(-1, 1)\n\npr
ice_scalar = StandardScaler()\nprice_scalar.fit(project_data[\'price\'].va
lues.reshape(-1,1)) # finding the mean and standard deviation of this data
\nprint(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(pr
ice_scalar.var_[0])}")\n\n# Now standardize the data with above maen and v
ariance.\nprice_standardized = price_scalar.transform(project_data[\'price
\'].values.reshape(-1, 1))\n\n'
```

In [62]:

```
# price_standardized
```

### 1.5.4 Merging all the above features

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

In [63]:

```
'''
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
'''
```

Out[63]:

```
'\nprint(categories_one_hot.shape)\nprint(sub_categories_one_hot.shape)\np
rint(text_bow.shape)\nprint(price_standardized.shape)\n'
```

In [64]:

```
'''
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
'''
```

Out[64]:

```
'\n# merge two sparse matrices: https://stackoverflow.com/a/19710648/40840
39\nfrom scipy.sparse import hstack\n# with the same hstack function we ar
e concatinating a sparse matrix and a dense matirx :)\nX = hstack((categor
ies_one_hot, sub_categories_one_hot, text_bow, price_standardized))\nX.sha
pe\n'
```

# Assignment 3: Apply KNN

1. **[Task-1] Apply KNN(brute force version) on these feature sets**

   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)
   - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)
   - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

2. **Hyper paramter tuning to find best K**

   - Find the best hyper parameter which results in the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
   - Find the best hyper paramter using k-fold cross validation (or) simple cross validation data
   - Use gridsearch-cv or randomsearch-cv or write your own for loops to do this task

3. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure 
   - Once you find the best hyper parameter, you need to train your model-M using the best hyper-param. Now, find the AUC on test data and plot the ROC curve on both train and test using model-M.

     
   - Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points

4. **[Task-2]**

   - Select top 2000 features from feature Set 2 using `SelectKBest` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html) and then apply KNN on top of these features
   - ```
     from sklearn.datasets import load_digits
     from sklearn.feature_selection import SelectKBest, chi2
     X, y = load_digits(return_X_y=True)
     X.shape
     X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
     X_new.shape
     ========
     output:
     (1797, 64)
     (1797, 20)
     ```
   - Repeat the steps 2 and 3 on the data matrix after feature selection

5. **Conclusion**

   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link (http://zetcode.com/python/prettytable/)
     ```

**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. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

# 2. K Nearest Neighbor

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

In [65]:

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

In [66]:

```
project_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 109248 entries, 0 to 109247
Data columns (total 20 columns):
 #   Column                                          Non-Null Count   Dtype
---  ------                                          --------------   -----
 0   Unnamed: 0                                      109248 non-null  int64
 1   id                                              109248 non-null  object
 2   teacher_id                                      109248 non-null  object
 3   teacher_prefix                                  109248 non-null  object
 4   school_state                                    109248 non-null  object
 5   Date                                            109248 non-null  dateti
me64[ns]
 6   project_grade_category                          109248 non-null  object
 7   project_title                                   109248 non-null  object
 8   project_essay_1                                 109248 non-null  object
 9   project_essay_2                                 109248 non-null  object
 10  project_essay_3                                 3758 non-null    object
 11  project_essay_4                                 3758 non-null    object
 12  project_resource_summary                        109248 non-null  object
 13  teacher_number_of_previously_posted_projects    109248 non-null  int64
 14  project_is_approved                             109248 non-null  int64
 15  clean_categories                                109248 non-null  object
 16  clean_subcategories                             109248 non-null  object
 17  essay                                           109248 non-null  object
 18  price                                           109248 non-null  float6
4
 19  quantity                                        109248 non-null  int64
dtypes: datetime64[ns](1), float64(1), int64(4), object(14)
memory usage: 17.5+ MB
```

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

## Dropping the unnecessary columns

In [67]:

```
data1 = project_data.drop(['Unnamed: 0', 'id','Date','project_essay_1','project_essay_
2','project_essay_3','project_essay_4','project_resource_summary','teacher_id','quantit
y'], axis = 1)
```

In [68]:

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 109248 entries, 0 to 109247
Data columns (total 10 columns):
 #   Column                                        Non-Null Count   Dtyp
e
---  ------                                        --------------   ----
-
 0   teacher_prefix                                109248 non-null  obje
ct
 1   school_state                                  109248 non-null  obje
ct
 2   project_grade_category                        109248 non-null  obje
ct
 3   project_title                                 109248 non-null  obje
ct
 4   teacher_number_of_previously_posted_projects  109248 non-null  int6
4
 5   project_is_approved                           109248 non-null  int6
4
 6   clean_categories                              109248 non-null  obje
ct
 7   clean_subcategories                           109248 non-null  obje
ct
 8   essay                                         109248 non-null  obje
ct
 9   price                                         109248 non-null  floa
t64
dtypes: float64(1), int64(2), object(7)
memory usage: 9.2+ MB
```

In [69]:

```
data1 = data1[:50000]
```

In [ ]:

In [71]:

```
y = data1['project_is_approved'].values
X = data1.drop(['project_is_approved'], axis=1)
X.head(1)
```

Out[71]:

| | teacher_prefix | school_state | project_grade_category | project_title | teacher_number_of_previ |
|---|---|---|---|---|---|
| 0 | Mrs. | CA | Grades PreK-2 | Engineering STEAM into the Primary Classroom | |

In [72]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, strat
ify=y_train)
```

In [ ]:

```

```

# 2.2 Make Data Model Ready: encoding numerical, categorical features

In [73]:

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

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

## 2.2.1 Numerical features

1. teacher_number_of_previously_posted_projects
2. price

#### *2.2.1.1 Teacher number of previously posted projects*

In [74]:

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

X_train_TPPP_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_cv_TPPP_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_test_TPPP_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_TPPP_norm.shape, y_train.shape)
print(X_cv_TPPP_norm.shape, y_cv.shape)
print(X_test_TPPP_norm.shape, y_test.shape)
print("="*100)
```

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

In [75]:

```
print("Transpose of teacher number of previously posted projects")

X_train_TPPP_norm = X_train_TPPP_norm.transpose()
X_cv_TPPP_norm = X_cv_TPPP_norm.transpose()
X_test_TPPP_norm = X_test_TPPP_norm.transpose()

print("After transpose")
print(X_train_TPPP_norm.shape, y_train.shape)
print(X_cv_TPPP_norm.shape, y_cv.shape)
print(X_test_TPPP_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of teacher number of previously posted projects
After transpose
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

#### *2.2.1.2 price*

In [76]:

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

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(1,-1))
X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(1,-1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

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

In [77]:

```
print("Transpose of price")

X_train_price_norm = X_train_price_norm.transpose()
X_cv_price_norm = X_cv_price_norm.transpose()
X_test_price_norm = X_test_price_norm.transpose()


print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of price
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
==========================
```

## 2.2.2 Categorical Data

#### Categorical Features for vectorization

1. Clean Categories
2. Clean Sub Categories
3. School State
4. Teacher Prefix
5. Project grade category

#### *2.2.2.1 Clean Categories*

In [96]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train da
ta

# we use the fitted CountVectorizer to convert the text to vector
X_train_CC_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_cv_CC_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_CC_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_CC_ohe.shape, y_train.shape)
print(X_cv_CC_ohe.shape, y_cv.shape)
print(X_test_CC_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
====================================================================================
==========================
```

#### *2.2.2.2 Clean Sub Categories*

In [97]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal
se, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train
data

# we use the fitted CountVectorizer to convert the text to vector
X_train_CSC_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_cv_CSC_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_CSC_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_CSC_ohe.shape, y_train.shape)
print(X_cv_CSC_ohe.shape, y_cv.shape)
print(X_test_CSC_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutrition
Education', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'A
ppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Lit
eracy']
====================================================================================
==========================
```

#### *2.2.2.3 School State*

In [95]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_scl_dict.keys()), lowercase=False,
binary=True)
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME',
'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'N
V', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'M
A', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'N
Y', 'TX', 'CA']
====================================================================================
==========================
```

#### *2.2.2.4 Teacher prefix*

In [94]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_tp_dict.keys()), lowercase=False, b
inary=True)
vectorizer.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 5) (22445,)
(11055, 5) (11055,)
(16500, 5) (16500,)
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
====================================================================================
==========================
```

#### *2.2.2.5 Project Grade category*

In [98]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_pgc_dict.keys()), lowercase=False,
binary=True)
vectorizer.fit(X_train['project_grade_category'].values) # fit has to happen only on tr
ain data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['project_grade_category'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
['9-12', '6-8', '3-5', 'PreK-2']
====================================================================================
==========================
```

In [99]:

```
data1['project_grade_category'].unique()
```

Out[99]:

```
array(['Grades PreK-2', 'Grades 3-5', 'Grades 9-12', 'Grades 6-8'],
      dtype=object)
```

## 2.3 Make Data Model Ready: encoding eassay, and project_title

In [100]:

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

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

**Ecoding Essay and Project title**



## 2.3.1 BOW Essays and Title

### *2.3.1.1 BOW Essay*

In [101]:

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

print("="*100)


vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer.transform(X_train['essay'].values)
X_cv_essay_bow = vectorizer.transform(X_cv['essay'].values)
X_test_essay_bow = vectorizer.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```

```
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
====================================================================================================
After vectorizations
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
====================================================================================================
```

### *2.3.1.2 BOW Title*

In [102]:

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

print("="*100)


vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_bow = vectorizer.transform(X_train['project_title'].values)
X_cv_title_bow = vectorizer.transform(X_cv['project_title'].values)
X_test_title_bow = vectorizer.transform(X_test['project_title'].values)

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
print("="*100)
```

```
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
====================================================================================
==========================
After vectorizations
(22445, 2645) (22445,)
(11055, 2645) (11055,)
(16500, 2645) (16500,)
====================================================================================
==========================
```

## 2.3.2 TF IDF Essay and Title

#### *2.3.2.1 TF IDF Essay*

In [103]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

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

print("="*100)


vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].values)
X_cv_essay_tfidf = vectorizer.transform(X_cv['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

```
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
====================================================================================
==========================
After vectorizations
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
====================================================================================
==========================
```

#### *2.3.2.2 TF IDF Title*

In [104]:

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

print("="*100)


vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_tfidf = vectorizer.transform(X_train['project_title'].values)
X_cv_title_tfidf = vectorizer.transform(X_cv['project_title'].values)
X_test_title_tfidf = vectorizer.transform(X_test['project_title'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print("="*100)
```

```
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
====================================================================================
==========================
After vectorizations
(22445, 2645) (22445,)
(11055, 2645) (11055,)
(16500, 2645) (16500,)
====================================================================================
==========================
```

## 2.3.3 AVG W2V Essay and Title

#### *2.3.3.1 AVG W2V Essay*

In [105]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-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())
```

In [106]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(X_train['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

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

100%|██████████| 22445/22445 [00:28<00:00, 800.37it/s]

22445
300
[-1.27352950e-02 -7.26544768e-02 -1.43440947e-02 -2.06926755e-01
  8.18185497e-02 -5.27883177e-02 -3.64595672e+00  2.13579024e-01
  2.95227863e-03 -2.25145503e-01  1.16549708e-01  1.55926208e-02
  5.62802137e-02 -6.73401164e-02 -1.04385680e-01 -9.62916023e-02
 -1.14269515e-01 -5.07839458e-02  9.70147523e-02  9.46668458e-02
 -5.82453721e-03 -3.00802288e-02 -7.97175621e-02  7.10117073e-02
 -1.10311051e-02 -2.81808737e-02  5.01527832e-02 -1.56169432e-01
 -8.97351961e-02 -9.66535511e-02 -2.75064854e-01 -4.16095469e-02
  6.70259179e-02 -4.56709324e-02 -1.54017775e-01 -6.09757656e-02
 -4.97772752e-02 -1.14521398e-01  2.62904136e-02 -7.30896046e-02
 -4.71173519e-02  1.23115756e-01  1.73145447e-02 -2.26032098e-01
 -6.63557588e-02 -8.44574000e-02  5.17626286e-02 -1.37914229e-01
 -8.21359576e-02 -4.66357885e-02  5.30882106e-02  6.01884847e-02
 -5.35435240e-02 -7.96705420e-03  4.96984525e-02 -9.65536760e-02
  1.05209562e-01 -1.49069893e-02 -1.22499874e-01  1.03425203e-01
 -2.20119721e-02  2.76479771e-03  2.71988492e-02 -6.86049576e-02
 -1.44694275e-03  1.17485974e-01  4.52585590e-02  5.13235015e-02
  2.30208760e-01 -1.47692915e-01 -1.56727059e-01  4.81655962e-02
 -3.03959279e-02 -6.89056057e-02 -2.98821455e-02 -2.21777593e-01
  1.36859307e-01  5.76832617e-02  7.90144943e-02 -8.35581970e-02
  2.88586351e-02 -3.84410789e-01 -7.37268939e-02 -1.30958639e-01
 -4.11100637e-02  6.05760205e-02  7.79416874e-02 -1.27427682e-01
  8.46257168e-02 -3.13491679e-02  2.62551842e-02 -2.13261221e-02
 -4.48375576e-02  1.04841129e-01  5.33540529e-02 -2.98383794e-01
 -2.46546935e+00 -4.59156375e-02  1.26491460e-01  8.78291584e-02
 -7.32378729e-02  9.60512080e-02  2.70633794e-01  3.50993031e-02
 -3.19637248e-02  2.08372362e-02  7.12718691e-02 -1.48547297e-01
 -2.27708932e-02 -2.65889634e-02 -4.01731445e-02  2.94970291e-02
  4.30832733e-02  1.54657741e-01 -8.88752847e-02  5.02005431e-02
 -9.23378347e-02 -2.45359405e-02  1.10004081e-01  6.77325905e-02
 -6.40610293e-02 -1.22962531e-02  7.24893561e-02 -1.98103901e-01
  1.06237511e-02  3.68264885e-03  6.35475529e-02 -5.89031439e-02
 -1.28746815e-02  1.29472338e-01  1.14981339e-02  4.23692492e-02
 -4.06709122e-02 -8.61816485e-02  8.26334233e-02  1.63106489e-02
  5.98686004e-02  2.87173500e-03  2.21025905e-01  3.01091768e-01
  1.08361131e-01  1.58247829e-02  5.06845603e-02  1.96263321e-02
 -3.61771783e-02 -2.86459653e-02  6.04106427e-02 -5.62650985e-02
  3.05024165e-01  1.24125658e-01 -1.55692847e-02 -8.01591958e-02
  4.47069769e-02 -6.26638401e-02  4.01689015e-02 -6.36379565e-02
 -9.87068435e-03  6.63663550e-03 -1.27201039e-01 -6.73536336e-03
  1.06022422e-01 -8.40344878e-02 -1.94087473e-02 -4.88515523e-02
 -4.57271172e-02  2.66509352e-02 -3.89819656e-02  1.29967887e-01
  1.64647170e-01 -7.61099359e-02 -2.27288802e-02 -3.50758118e-02
 -1.01943588e-01 -9.32858866e-02 -1.12592627e-01  2.02575327e-01
 -5.06574018e-02 -7.13511683e-02 -4.31552252e-02 -9.00224834e-02
  1.57552640e-01  1.19513642e-01 -4.44831176e-02 -7.11094481e-02
  1.75085706e-02 -1.36413635e-01 -2.92575992e-03  4.03371458e-02
  1.27464520e-01 -3.34870859e-02 -2.81577986e-02 -5.33371015e-02
 -3.81904892e-02 -1.01438482e-02  3.54576641e-02 -1.10097944e-01
  1.20115948e-01  7.08787508e-02  1.09718269e-01 -8.77800267e-04
  1.68753193e-01 -5.98295400e-02 -6.22390752e-03  2.74629879e-02
  5.25464389e-03  1.05456795e-01  4.03212615e-02 -6.53394309e-02
  1.87109636e-01 -7.38246649e-02  1.05900330e-01 -1.11501672e-02
 -1.03467487e-01 -1.14148700e-01 -3.05296401e-02 -5.56663053e-03
 -1.12862323e-01 -7.59497531e-02 -2.65960057e-02  4.12154286e-02
 -9.95588931e-02 -1.44141969e-01 -3.53871336e-02  1.25887937e-02
 -2.63991446e+00  1.13048157e-01 -2.43535764e-02  2.84285382e-02
  5.33384084e-03 -1.03127013e-01  7.42570882e-02  6.37780575e-02
 -1.88237031e-02 -1.96170008e-02 -1.17239901e-01  1.56466515e-01

```
 8.65311397e-03 -4.52477240e-02 -5.88912099e-02  4.44335317e-02
-1.37988206e-01  6.21087691e-02 -1.54858138e-01  6.63943176e-02
-1.76055588e-02 -6.43274420e-02 -3.79844370e-02 -5.54937976e-02
-1.98026650e-02  3.76965680e-02 -2.56596011e-02 -5.33947439e-02
 1.92730179e-02 -5.02899205e-02  9.64769156e-02  1.64874061e-02
 5.88088196e-02 -6.61738683e-02 -8.46449924e-03 -3.24361473e-02
 9.23456305e-02 -1.29264485e-02 -5.41690996e-02 -4.09634996e-02
 1.24451300e-01 -1.15777509e-01 -2.62408160e-01 -6.94021615e-02
 1.05463255e-01  3.79208809e-02 -3.46448763e-02 -1.39178910e-01
-1.88795864e-01  4.28031546e-02  1.97419618e-03  8.66931313e-02
 3.45605485e-02  4.97563597e-02 -4.54409385e-02 -1.65834150e-02
 3.70230128e-01 -3.26235821e-02  2.90181912e-02  1.38245948e-01
-1.53698275e-02  1.40294103e-01  1.14695935e-02  3.07665729e-02
-2.78887042e-02 -5.08775382e-02 -1.33081279e-02 -8.84683382e-02
-7.38555649e-02 -4.96375233e-02 -4.91424198e-03  5.16252137e-02
-6.00056217e-02  2.42709311e-02  9.40706852e-02  7.55543359e-03]
```

In [107]:

```
avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)
```

```
100%|██████████| 11055/11055 [00:13<00:00, 795.27it/s]
```

In [108]:

```
avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_test['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)
```

```
100%|██████████| 16500/16500 [00:16<00:00, 999.03it/s]
```

#### *2.3.3.2 AVG W2V Title*

In [109]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train_title = []; # the avg-w2v for each sentence/review is stored in t
his list
for sentence in tqdm(X_train['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train_title.append(vector)

print(len(avg_w2v_vectors_train_title))
print(len(avg_w2v_vectors_train_title[0]))
print(avg_w2v_vectors_train_title[0])
```

100%|██████████| 22445/22445 [00:00<00:00, 55785.20it/s]

22445
300
[-9.6110e-02 -2.5788e-01 -3.5860e-01 -3.2887e-01  5.7950e-01 -5.1774e-01
 -4.1582e+00 -1.1371e-01 -1.0848e-01 -4.8885e-01  1.9931e-01 -1.0540e-01
 -4.3825e-01 -3.4483e-01 -4.5052e-01 -3.4864e-01 -4.5800e-01 -8.1554e-01
  2.2006e-01  2.0254e-01 -1.0954e-01  1.2520e-01 -5.4117e-01  3.4731e-01
 -9.9998e-02 -1.8998e-02 -1.4277e-01 -4.2481e-01 -9.4091e-03 -4.3155e-01
 -3.8769e-02  1.2147e-01  5.1988e-01 -4.9840e-01 -2.4625e-01 -5.2067e-01
 -5.8210e-02 -3.0712e-01  2.5512e-01  4.8033e-02 -2.2313e-01 -6.9182e-03
  3.9824e-02 -5.0088e-01 -1.1972e-01 -7.9045e-02  1.6880e-02 -3.4052e-01
 -2.0660e-01  8.1265e-02  1.2352e-01 -4.9007e-01  3.4946e-01 -2.9241e-01
  1.4893e-01  1.3660e-01 -9.7830e-02 -6.8472e-02 -1.0913e-02  2.8454e-03
 -1.2656e-01  3.4270e-01  1.0580e-01 -4.6151e-01  7.0133e-02 -6.1343e-02
 -1.5021e-02  1.7659e-01  1.7941e-01 -5.1377e-01 -3.1381e-01 -1.3720e-01
  4.5186e-02 -8.2259e-02  2.1515e-01 -2.1955e-01  1.0313e-01 -2.0704e-01
  1.4041e-01 -3.5151e-01  6.2316e-01 -5.7990e-01 -5.6115e-02 -2.1746e-03
  1.8958e-01  2.2398e-01  1.2246e-01 -2.6178e-01  1.0779e-02 -3.1268e-01
 -2.1447e-01  3.5344e-01 -2.6041e-02  1.8232e-02  3.5751e-01 -7.0188e-02
 -3.0872e+00 -1.3131e-01  1.7387e-02  2.3244e-01 -6.0585e-02  2.0679e-01
  5.7579e-01  3.6338e-01 -4.1574e-01  3.0607e-02  2.3619e-01 -1.1284e-01
 -3.6043e-01  2.1635e-01 -2.7520e-02  1.7502e-01  4.3491e-01 -8.8247e-02
  4.0754e-01 -4.8551e-01  1.3539e-01 -9.0759e-02  1.4423e-01  3.4118e-01
 -3.7940e-01 -2.7344e-01  2.5930e-02  7.3217e-02 -1.0176e-01  1.6551e-01
 -2.3278e-01 -1.8563e-01  2.1372e-02 -9.3111e-02  1.5179e-01  1.5057e-01
  5.5148e-01 -2.0088e-01 -7.9495e-02  2.2599e-01  2.6243e-01  2.5123e-01
  6.0266e-01 -2.0423e-01  3.6972e-01 -1.0694e-01  7.2887e-03  1.8359e-02
  2.2368e-01 -1.4065e-01  1.1120e-01  8.7667e-02  8.4660e-01  3.1545e-01
 -1.5348e-01  2.0311e-02  2.0878e-02  3.8651e-01  4.7422e-02 -2.4854e-01
 -1.9053e-01  4.9173e-01  3.8161e-02 -2.1038e-02  1.4496e-01  1.1591e-01
 -1.5105e-01 -1.8942e-01  1.8703e-01  2.6752e-02  4.6523e-03  3.9814e-01
 -1.8617e-02 -7.3177e-01  7.2832e-02  4.1535e-01 -4.8818e-01  3.0400e-03
 -2.2729e-01  8.8248e-01 -6.1612e-01 -1.8901e-01 -3.3491e-01 -2.8672e-01
 -1.3143e-02 -3.7545e-01 -1.8443e-01 -5.5218e-01  7.0186e-01 -7.3107e-02
  6.3930e-01  1.3098e-01  7.1586e-02  5.3641e-03  2.4636e-01 -7.0744e-01
 -4.5036e-01  6.0187e-04 -3.9093e-01 -2.7160e-02  2.5589e-01 -1.7313e-01
  2.9883e-01 -9.0947e-04  8.3140e-02 -4.0990e-01 -1.3024e-02 -4.9533e-02
  3.0410e-01  6.4302e-01  2.3045e-01 -1.8757e-01  3.7584e-02 -2.6082e-01
  1.7530e-01 -6.2815e-02 -2.2569e-01 -1.2130e-01  1.5524e-01 -1.4407e-01
  8.8732e-02  3.4674e-01 -4.3494e-01  3.8688e-01 -1.5733e-01 -1.2721e-01
  3.0194e-01  3.2034e-01 -3.3264e+00  6.9427e-02  1.3848e-01 -5.8216e-02
 -2.7088e-02  1.1028e-01  3.4040e-01  1.8654e-01  1.1522e-01 -4.0381e-01
  4.4776e-02  1.5535e-01  1.6247e-01 -2.4051e-01  4.7290e-02  3.4980e-02
 -7.5942e-02  1.5598e-01 -5.9873e-02  4.6743e-03  1.5595e-01 -2.7613e-01
  1.3562e-01  1.3485e-01 -7.3724e-02  3.1421e-01  3.1234e-02 -2.3516e-01
  3.1005e-01 -1.0375e-01 -3.0783e-01 -5.5327e-01  2.8304e-01  8.1429e-02
  3.7778e-01  1.5725e-01  1.1757e-02  4.3006e-02 -4.3423e-01 -2.2718e-01
 -4.3292e-02 -6.3617e-01 -8.9390e-01 -1.7406e-01  4.1111e-01 -1.4404e-01
 -1.6780e-01 -4.4438e-01 -7.3051e-01  1.0957e-01  1.3122e-01  8.5623e-02
  1.2504e-01 -4.0337e-01  4.1765e-02 -2.7574e-01  6.2513e-02  5.1093e-02
  3.9926e-01  1.1149e-01 -5.6462e-02  2.6809e-01 -3.9569e-01  3.1033e-01
 -4.9750e-02 -3.3139e-01  4.7781e-01 -2.1213e-02 -2.1236e-01  4.2374e-01
  1.4083e-01  6.7498e-02 -1.2675e-01 -3.7030e-01 -9.2774e-02  3.9058e-01]

In [110]:

```
avg_w2v_vectors_cv_title = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(X_cv['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv_title.append(vector)
```

```
100%|██████████| 11055/11055 [00:00<00:00, 49569.58it/s]
```

In [111]:

```
avg_w2v_vectors_test_title = []; # the avg-w2v for each sentence/review is stored in th
is list
for sentence in tqdm(X_test['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test_title.append(vector)
```

```
100%|██████████| 16500/16500 [00:00<00:00, 57342.44it/s]
```

## 2.3.4 TF IDF W2V Essay and Title

#### *2.3.4.1 TF IDF W2V Essay*

In [112]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'].values)
# 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 [113]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this
 list
for sentence in tqdm(X_train['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_train.append(vector)

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

```
100%|██████████| 22445/22445 [04:18<00:00, 86.84it/s]

22445
300
```

In [114]:

```
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_cv['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_cv.append(vector)
```

```
100%|██████████| 11055/11055 [01:36<00:00, 114.42it/s]
```

In [115]:

```
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(X_test['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_test.append(vector)
```

```
100%|██████████| 16500/16500 [02:52<00:00, 95.44it/s]
```

#### *2.3.4.2 TF IDF W2V Title*

In [116]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['project_title'].values)
# 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 [117]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train_title = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(X_train['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_train_title.append(vector)

print(len(tfidf_w2v_vectors_train_title))
print(len(tfidf_w2v_vectors_train_title[0]))
```

```
100%|██████████| 22445/22445 [00:00<00:00, 43256.22it/s]

22445
300
```

In [118]:

```
tfidf_w2v_vectors_cv_title = []; # the avg-w2v for each sentence/review is stored in th
is list
for sentence in tqdm(X_cv['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_cv_title.append(vector)
```

```
100%|██████████| 11055/11055 [00:00<00:00, 38650.17it/s]
```

In [119]:

```
tfidf_w2v_vectors_test_title = []; # the avg-w2v for each sentence/review is stored in
 this list
for sentence in tqdm(X_test['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_test_title.append(vector)
```

```
100%|██████████| 16500/16500 [00:00<00:00, 45951.09it/s]
```

In [ ]:

```

```

## Concatinating all the features

### *1. SET 1 BOW*

In [120]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_BOW = hstack((X_train_essay_bow, X_train_title_bow, X_train_state_ohe, X_train_tea
cher_ohe, X_train_grade_ohe, X_train_CSC_ohe, X_train_CC_ohe, X_train_price_norm, X_tra
in_TPPP_norm)).tocsr()
X_cr_BOW = hstack((X_cv_essay_bow, X_cv_title_bow, X_cv_state_ohe, X_cv_teacher_ohe, X_
cv_grade_ohe, X_cv_CSC_ohe, X_cv_CC_ohe, X_cv_price_norm, X_cv_TPPP_norm)).tocsr()
X_te_BOW = hstack((X_test_essay_bow, X_test_title_bow, X_test_state_ohe, X_test_teacher
_ohe, X_test_grade_ohe, X_test_CSC_ohe, X_test_CC_ohe, X_test_price_norm, X_test_TPPP_n
orm)).tocsr()

print("Final Data matrix")
print(X_tr_BOW.shape, y_train.shape)
print(X_cr_BOW.shape, y_cv.shape)
print(X_te_BOW.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 7746) (22445,)
(11055, 7746) (11055,)
(16500, 7746) (16500,)
====================================================================================
==========================
```

### *2. SET 2 TF IDF*

In [121]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_TFIDF = hstack((X_train_essay_tfidf, X_train_title_tfidf, X_train_state_ohe, X_tra
in_teacher_ohe, X_train_grade_ohe, X_train_CSC_ohe, X_train_CC_ohe, X_train_price_norm,
X_train_TPPP_norm)).tocsr()
X_cr_TFIDF = hstack((X_cv_essay_tfidf, X_cv_title_tfidf, X_cv_state_ohe, X_cv_teacher_o
he, X_cv_grade_ohe, X_cv_CSC_ohe, X_cv_CC_ohe, X_cv_price_norm, X_cv_TPPP_norm)).tocsr
()
X_te_TFIDF = hstack((X_test_essay_tfidf, X_test_title_tfidf, X_test_state_ohe, X_test_t
eacher_ohe, X_test_grade_ohe, X_test_CSC_ohe, X_test_CC_ohe, X_test_price_norm, X_test_
TPPP_norm)).tocsr()

print("Final Data matrix")
print(X_tr_TFIDF.shape, y_train.shape)
print(X_cr_TFIDF.shape, y_cv.shape)
print(X_te_TFIDF.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 7746) (22445,)
(11055, 7746) (11055,)
(16500, 7746) (16500,)
====================================================================================
==========================
```

### *3. SET 3 AVG W2V*

In [122]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_AVG_W2V = hstack((avg_w2v_vectors_train, avg_w2v_vectors_train_title, X_train_stat
e_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_CSC_ohe, X_train_CC_ohe, X_train
_price_norm, X_train_TPPP_norm)).tocsr()
X_cr_AVG_W2V = hstack((avg_w2v_vectors_cv, avg_w2v_vectors_cv_title, X_cv_state_ohe, X_
cv_teacher_ohe, X_cv_grade_ohe, X_cv_CSC_ohe, X_cv_CC_ohe, X_cv_price_norm, X_cv_TPPP_n
orm)).tocsr()
X_te_AVG_W2V = hstack((avg_w2v_vectors_test, avg_w2v_vectors_test_title, X_test_state_o
he, X_test_teacher_ohe, X_test_grade_ohe, X_test_CSC_ohe, X_test_CC_ohe, X_test_price_n
orm, X_test_TPPP_norm)).tocsr()

print("Final Data matrix")
print(X_tr_AVG_W2V.shape, y_train.shape)
print(X_cr_AVG_W2V.shape, y_cv.shape)
print(X_te_AVG_W2V.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 701) (22445,)
(11055, 701) (11055,)
(16500, 701) (16500,)
====================================================================================
==========================
```

***4. SET 4 TF IDF W2V***

In [123]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_TFIDF_W2V = hstack((tfidf_w2v_vectors_train, tfidf_w2v_vectors_train_title, X_trai
n_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_CSC_ohe, X_train_CC_ohe, X
_train_price_norm, X_train_TPPP_norm)).tocsr()
X_cr_TFIDF_W2V = hstack((tfidf_w2v_vectors_cv, tfidf_w2v_vectors_cv_title, X_cv_state_o
he, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_CSC_ohe, X_cv_CC_ohe, X_cv_price_norm, X_cv_
TPPP_norm)).tocsr()
X_te_TFIDF_W2V = hstack((tfidf_w2v_vectors_test, tfidf_w2v_vectors_test_title, X_test_s
tate_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_CSC_ohe, X_test_CC_ohe, X_test_p
rice_norm, X_test_TPPP_norm)).tocsr()

print("Final Data matrix")
print(X_tr_TFIDF_W2V.shape, y_train.shape)
print(X_cr_TFIDF_W2V.shape, y_cv.shape)
print(X_te_TFIDF_W2V.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 701) (22445,)
(11055, 701) (11055,)
(16500, 701) (16500,)
====================================================================================
==========================
```

In [ ]:

```

```

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

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

In [124]:

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

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

### 2.4.1 Applying KNN brute force on BOW, SET 1

In [125]:

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

In [126]:

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

    return y_data_pred
```

In [142]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
K = [5, 15,29,37,49,57,77]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_BOW, y_train)

    y_train_pred = batch_predict(neigh, X_tr_BOW)
    y_cv_pred = batch_predict(neigh, X_cr_BOW)

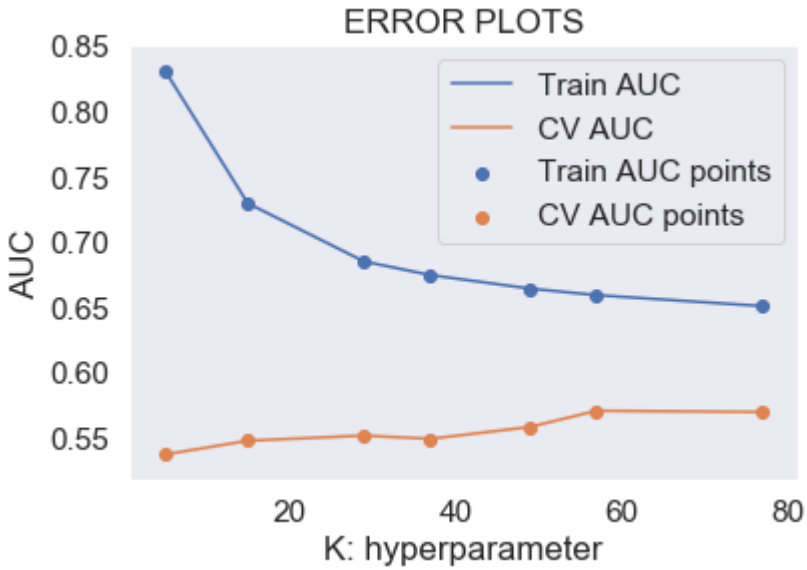
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

100%|██████████| 7/7 [10:10<00:00, 87.24s/it]



In [148]:

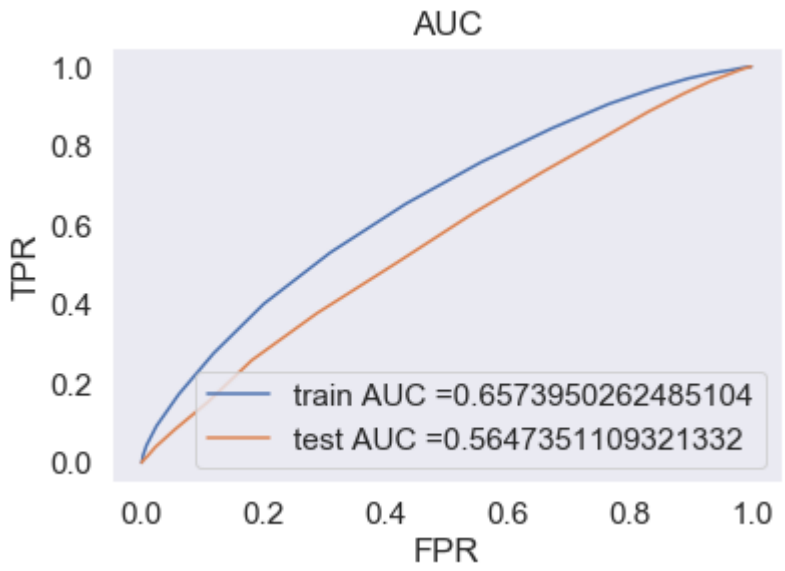
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=59, n_jobs=-1)
neigh.fit(X_tr_BOW, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr_BOW)
y_test_pred = batch_predict(neigh, X_te_BOW)

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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [149]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [150]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, trai
n_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tp
r)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.37042238889114343 for threshold 0.831
[[ 2037  1558]
 [ 6527 12323]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.29342479779495917 for threshold 0.847
[[1542 1100]
 [6891 6967]]
```

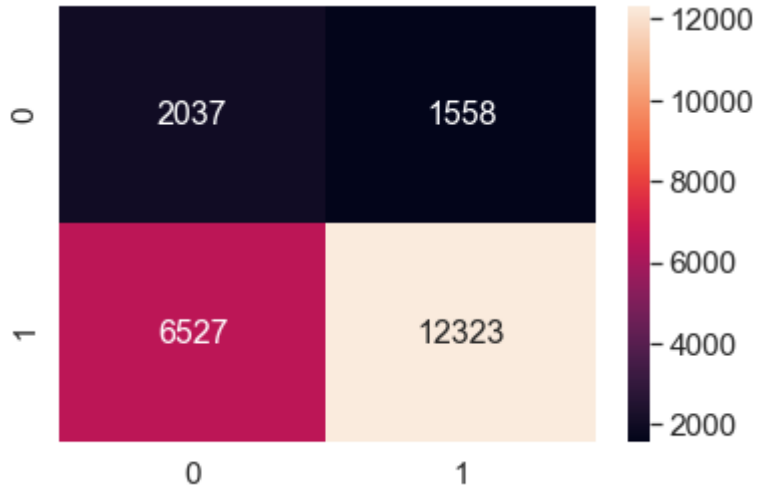
In [151]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, t
r_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.37042238889114343 for threshold 0.831
```

Out[151]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2993ca1b588>
```



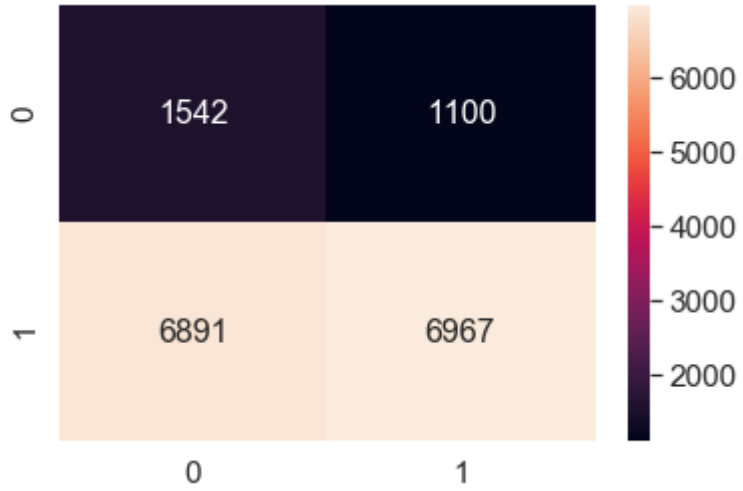
In [152]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_t
hresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.29342479779495917 for threshold 0.847
```

Out[152]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2995ce2efc8>
```




In [153]:

```
print(train_fpr.shape)
print(train_tpr.shape)
print(len(y_train_pred))
```

```
(26,)
(26,)
22445
```

In [ ]:

```

```

## 2.4.2 Applying KNN brute force on TFIDF, SET 2

In [134]:

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

In [135]:

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

    return y_data_pred
```

In [143]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
K = [15,37,49,57,69,77]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_TFIDF, y_train)

    y_train_pred = batch_predict(neigh, X_tr_TFIDF)
    y_cv_pred = batch_predict(neigh, X_cr_TFIDF)

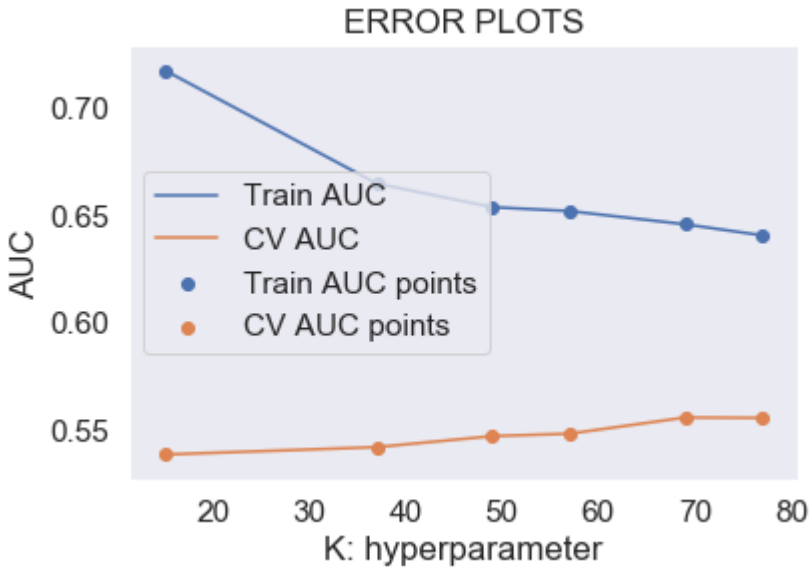
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

100%|██████████| 6/6 [08:27<00:00, 84.60s/it]



In [154]:

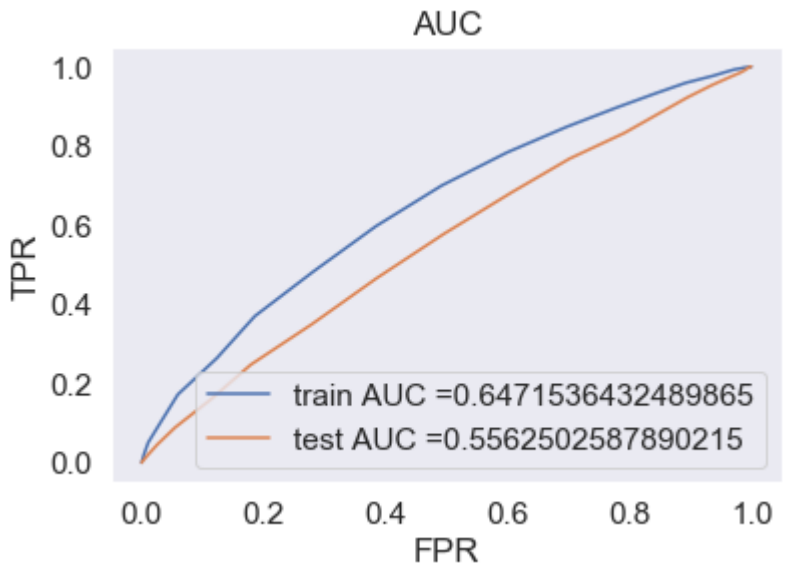
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=67, n_jobs=-1)
neigh.fit(X_tr_TFIDF, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr_TFIDF)
y_test_pred = batch_predict(neigh, X_te_TFIDF)

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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [155]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [156]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, trai
n_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tp
r)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.36764524852156144 for threshold 0.851
[[ 2212  1383]
 [ 7587 11263]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2913367322870045 for threshold 0.851
[[1334 1308]
 [5862 7996]]
```

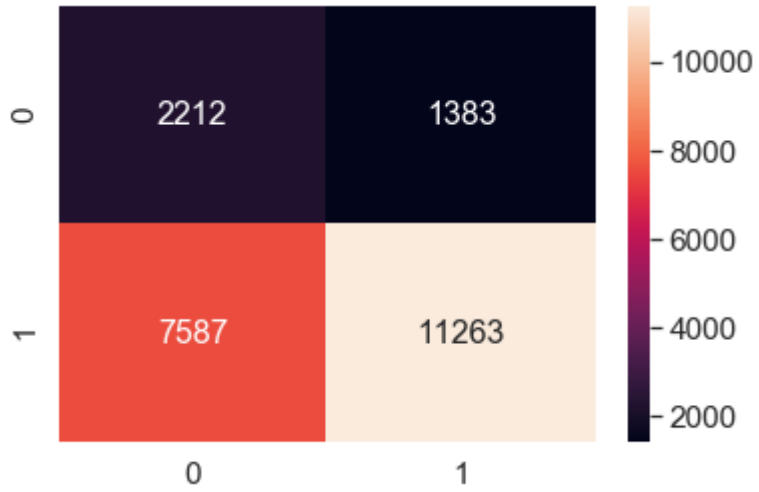
In [157]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, t
r_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.36764524852156144 for threshold 0.851
```

Out[157]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29913ccc0c8>
```



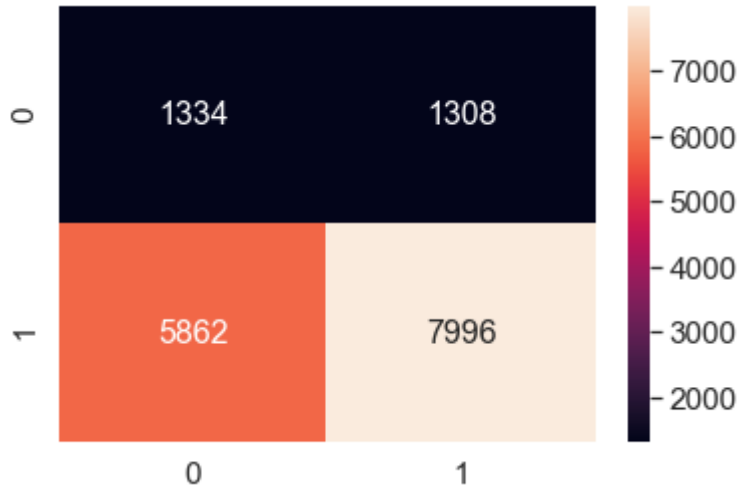
In [158]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_t
hresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2913367322870045 for threshold 0.851
```

Out[158]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2995cf788c8>
```




In [ ]:

```
```

### 2.4.3 Applying KNN brute force on AVG W2V, SET 3

In [154]:

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

In [ ]:

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

    return y_data_pred
```

In [144]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
K = [15,37,49,57,69,77]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_AVG_W2V, y_train)

    y_train_pred = batch_predict(neigh, X_tr_AVG_W2V)
    y_cv_pred = batch_predict(neigh, X_cr_AVG_W2V)

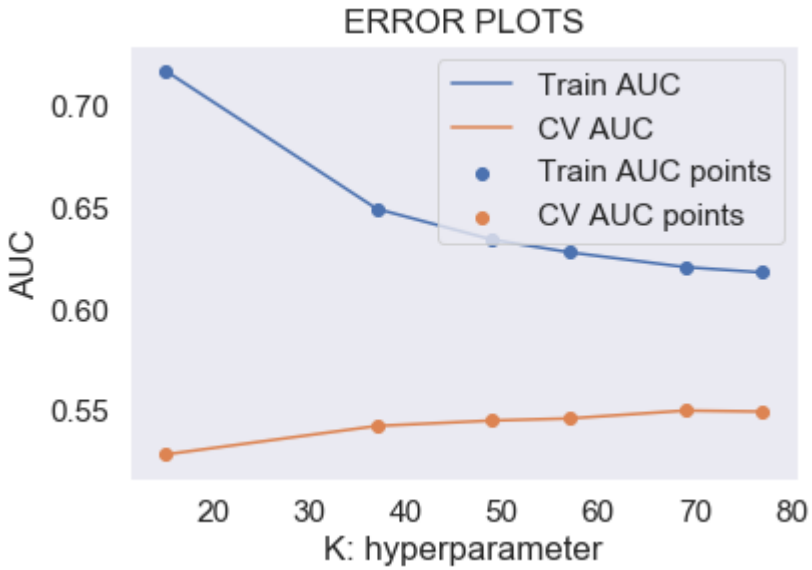
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

100%|██████████| 6/6 [34:01<00:00, 340.24s/it]



In [161]:

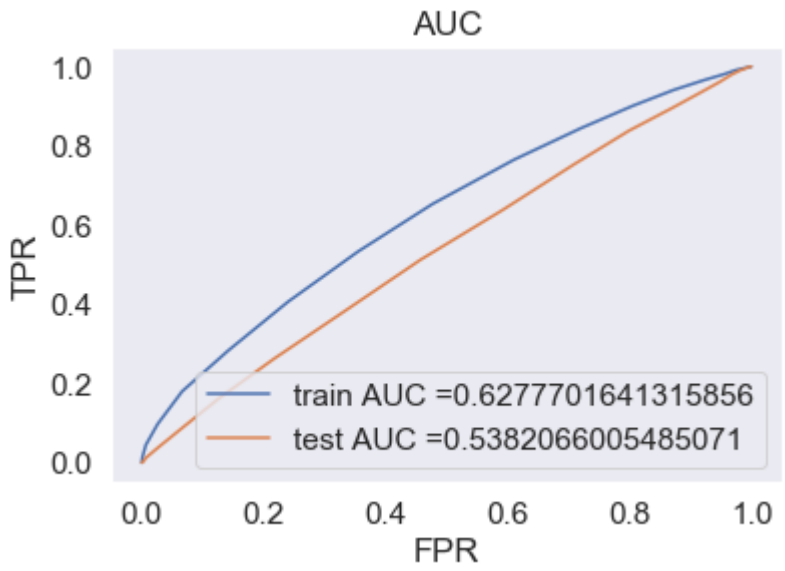
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=57, n_jobs=-1)
neigh.fit(X_tr_AVG_W2V, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr_AVG_W2V)
y_test_pred = batch_predict(neigh, X_te_AVG_W2V)

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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [162]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [163]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, trai
n_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tp
r)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.34386220177597093 for threshold 0.86
[[ 2320  1275]
 [ 8806 10044]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2779653015680075 for threshold 0.86
[[1434 1208]
 [6761 7097]]
```

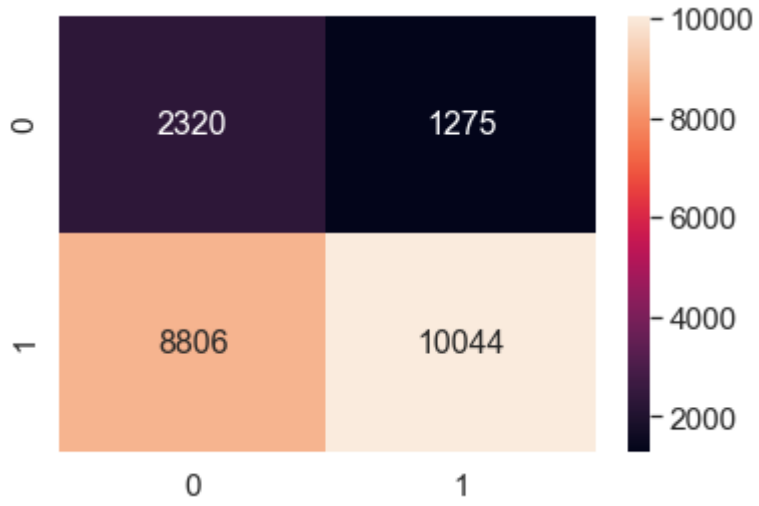
In [164]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, t
r_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.34386220177597093 for threshold 0.86
```

Out[164]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2995c534a08>
```



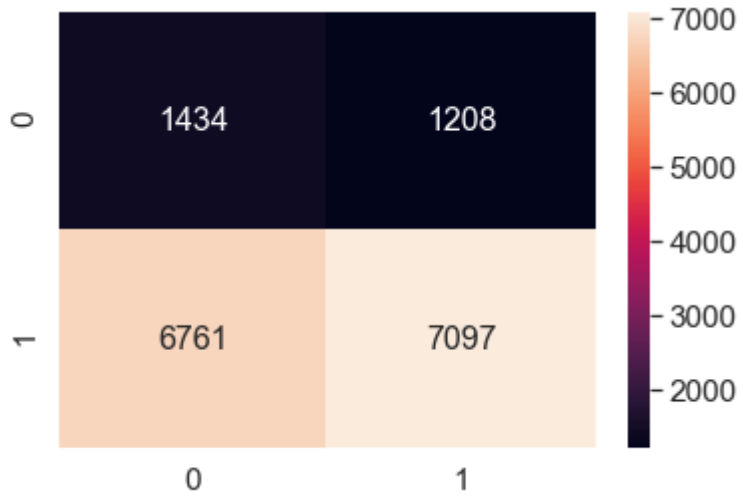
In [165]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_t
hresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2779653015680075 for threshold 0.86
```

Out[165]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x299173dde88>
```




In [ ]:

```

```

### 2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [162]:

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

In [163]:

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

    return y_data_pred
```

In [145]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
K = [15,37,49,57,69,77]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_TFIDF_W2V, y_train)

    y_train_pred = batch_predict(neigh, X_tr_TFIDF_W2V)
    y_cv_pred = batch_predict(neigh, X_cr_TFIDF_W2V)

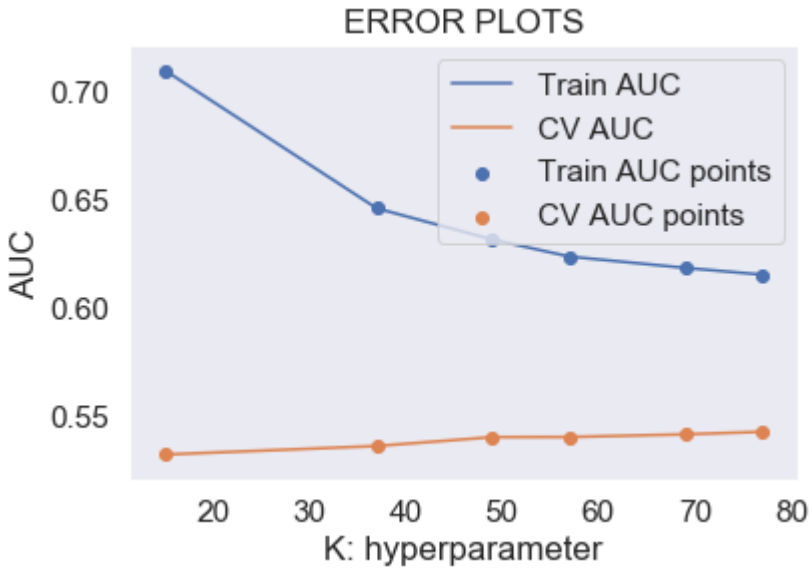
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

100%|██████████| 6/6 [34:31<00:00, 345.18s/it]



In [166]:

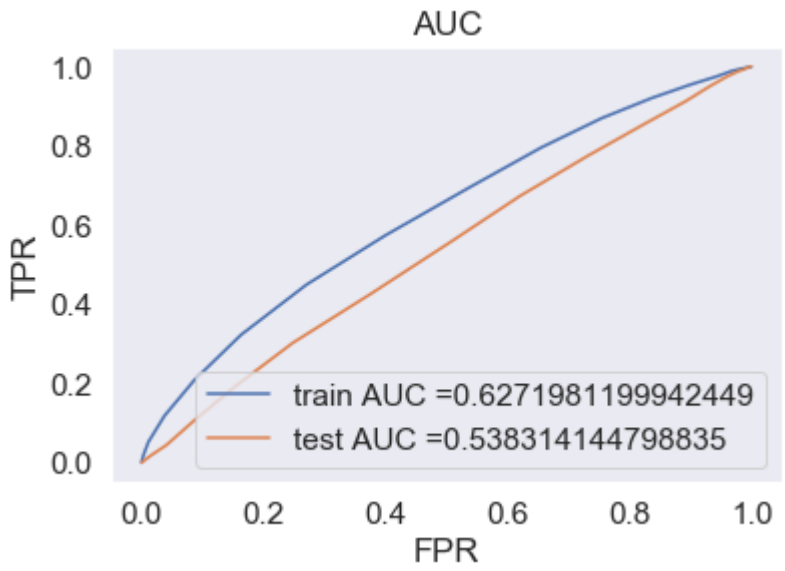
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=53, n_jobs=-1)
neigh.fit(X_tr_TFIDF_W2V, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr_TFIDF_W2V)
y_test_pred = batch_predict(neigh, X_te_TFIDF_W2V)

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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [167]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [168]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, trai
n_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tp
r)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.34449916071171643 for threshold 0.849
[[ 2158  1437]
 [ 8032 10818]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2750081419532756 for threshold 0.868
[[1647  995]
 [7940 5918]]
```

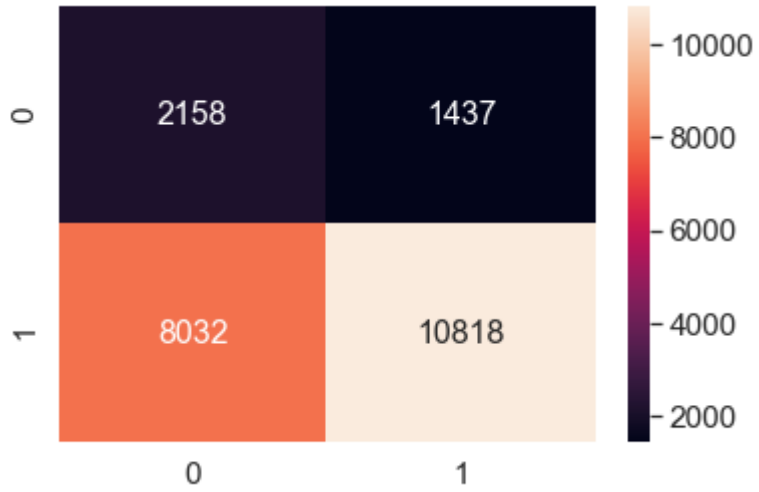
In [169]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, t
r_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.34449916071171643 for threshold 0.849
```

Out[169]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2995c2d30c8>
```



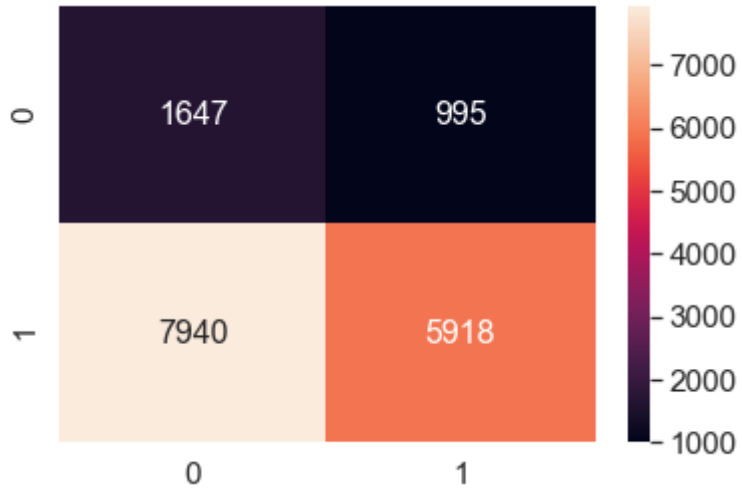
In [170]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_t
hresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2750081419532756 for threshold 0.868
```

Out[170]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29904f17f48>
```




In [ ]:

```

```

## 2.5 Feature selection with `SelectKBest`

In [171]:

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

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

In [172]:

```
from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2,k=2000).fit(X_tr_TFIDF, y_train)
X_tr_KBEST = t.transform(X_tr_TFIDF)
X_te_KBEST = t.transform(X_te_TFIDF)
X_cr_KBEST = t.transform(X_cr_TFIDF)

print("Final Data matrix on TFIDF")
print(X_tr_KBEST.shape, y_train.shape)
print(X_cr_KBEST.shape, y_cv.shape)
print(X_te_KBEST.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix on TFIDF
(22445, 2000) (22445,)
(11055, 2000) (11055,)
(16500, 2000) (16500,)
====================================================================================
==========================
```

In [173]:

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

    return y_data_pred
```

In [174]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence va
lues, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
K = [15,37,49,57,69,77]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_KBEST, y_train)

    y_train_pred = batch_predict(neigh, X_tr_KBEST)
    y_cv_pred = batch_predict(neigh, X_cr_KBEST)

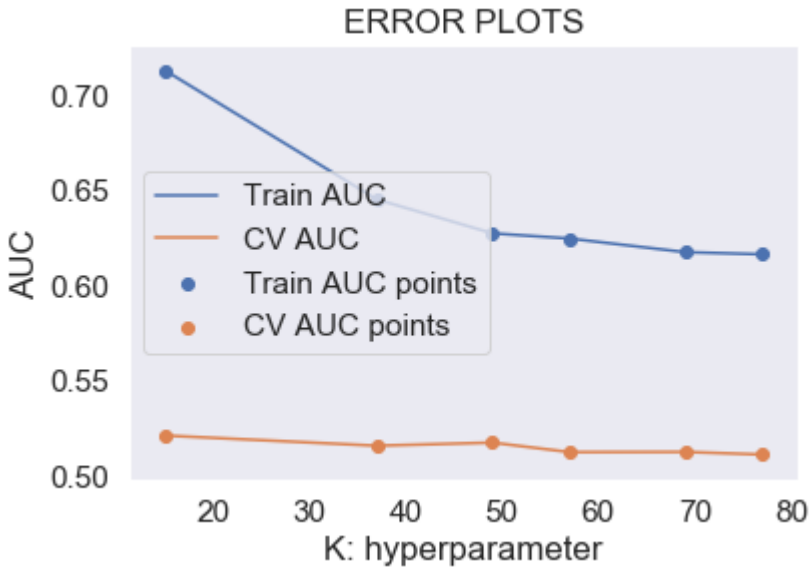
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

100%|██████████| 6/6 [04:45<00:00, 47.50s/it]



In [175]:

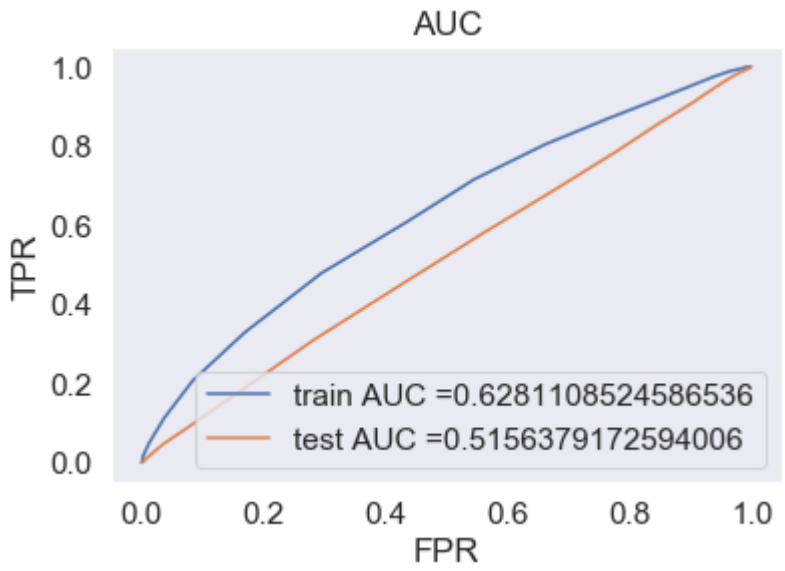
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=50, n_jobs=-1)
neigh.fit(X_tr_KBEST, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr_KBEST)
y_test_pred = batch_predict(neigh, X_te_KBEST)

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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [176]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [177]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, trai
n_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tp
r)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.3436759424930736 for threshold 0.84
[[ 2039  1556]
 [ 7428 11422]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25923422047939687 for threshold 0.86
[[1490 1152]
 [7488 6370]]
```

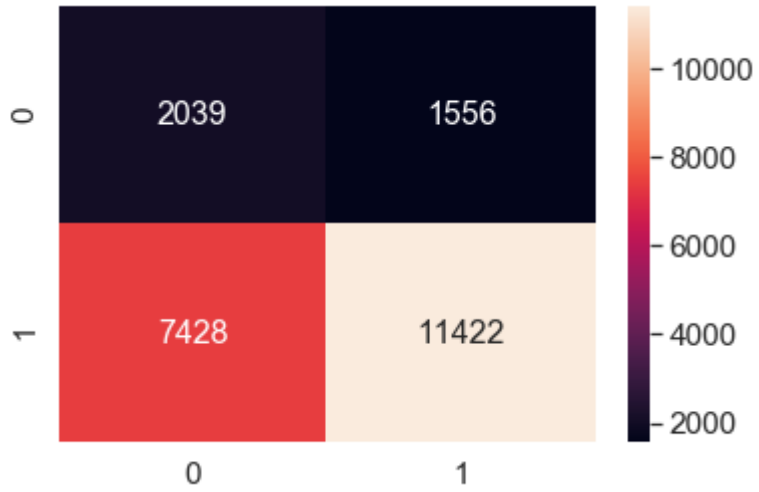
In [178]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, t
r_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.3436759424930736 for threshold 0.84
```

Out[178]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2995c97fcc8>
```



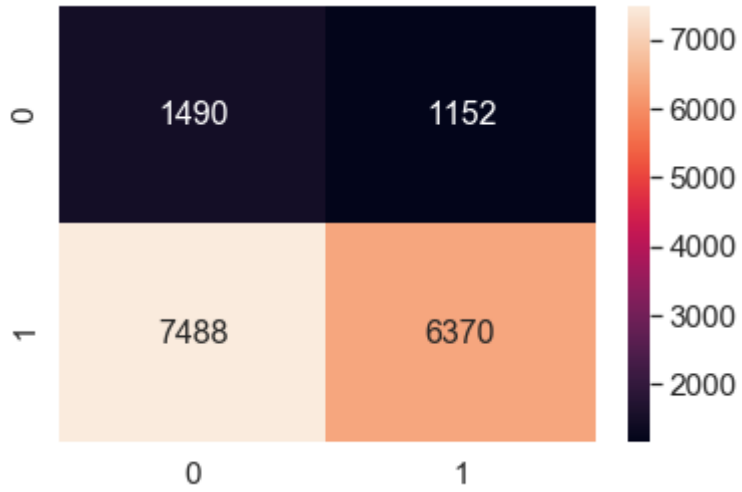
In [179]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_t
hresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.25923422047939687 for threshold 0.86
```

Out[179]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x29959f169c8>
```




In [ ]:

```
```

# 3. Conclusions

In [180]:

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

In [181]:

```
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prett
ytable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]
x.add_row(["BOW", "Brute", 59, 0.56])
x.add_row(["TFIDF", "Brute", 67, 0.55])
x.add_row(["AVG W2V", "Brute", 57, 0.53])
x.add_row(["TFIDF W2V", "Brute", 53, 0.53])
x.add_row(["TFIDF", "Top 2000", 50, 0.51])
print(x)
```

```
+------------+----------+-----------------+------+
| Vectorizer |  Model   | Hyper Parameter | AUC  |
+------------+----------+-----------------+------+
|    BOW     |  Brute   |        59       | 0.56 |
|   TFIDF    |  Brute   |        67       | 0.55 |
|  AVG W2V   |  Brute   |        57       | 0.53 |
| TFIDF W2V  |  Brute   |        53       | 0.53 |
|   TFIDF    | Top 2000 |        50       | 0.51 |
+------------+----------+-----------------+------+
```

## Observation

Among all the models, BOW has given the AUC value of 0.56. On the other hand, we can consider the top 2000/5000 features if the AUC difference is minimal in order to reduce the time complexity

In [ ]: