DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

Feature	Description
project_resource_summary	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
teacher_id	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
teacher_prefix	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [2]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [3]:

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

In [4]:

```
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 [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039

cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]
```

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


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


project_data.head(2)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5

In [6]:

```
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[6]:

	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

2. Preprocessing

2.1 preprocessing of `project_subject_categories`

In [7]:

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

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

```
        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]))
```

In [8]:

```
print(sorted_cat_dict)
```

```
{'Warmth': 1388, 'Care_Hunger': 1388, 'History_Civics': 5914, 'Music_Arts': 10293, 'AppliedLearning': 1
2135, 'SpecialNeeds': 13642, 'Health_Sports': 14223, 'Math_Science': 41421, 'Literacy_Language': 52239}
```

2.2 preprocessing of `project_subject_subcategories`

In [9]:

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

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunge
r"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=>
"Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e r
emoving 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>
"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    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]))
```

2.3 Text preprocessing of essay

In [10]:

```
# 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 [11]:

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

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5

In [12]:

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

In [13]:

```
# 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 classroom as well as the STEM journa
ls, which my students really enjoyed.  I would love to implement more of the Lakeshore STEM kits in my
classroom for the next school year as they provide excellent and engaging STEM lessons.My students come
from a variety of backgrounds, including language and socioeconomic status.  Many of them don't have a
lot of experience in science and engineering and these kits give me the materials to provide these exci
ting opportunities for my students.Each month I try to do several science or STEM/STEAM projects.  I wo
uld use the kits and robot to help guide my science 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 material in
the kits like tall tales (Paul Bunyan) or Johnny Appleseed.  The following units will be taught in the
next school year where I will implement these kits: magnets, motion, sink vs. float, robots.  I often g
et to these units and don't know If I am teaching the right way or using the right materials.    The ki
ts will give me additional ideas, strategies, and lessons to prepare my students in science.It is chall
enging to develop high quality science activities.  These kits give me the materials I need to provide
my students with science activities that will go along with the curriculum in my classroom.  Although I
have some things (like magnets) in my classroom, I don't know how to use them effectively.  The kits wi
ll provide 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 disabilities. My students all vary
in their ability level. However, the ultimate goal is to increase all students literacy levels. This in
cludes their reading, writing, and communication levels.I teach a really dynamic group of students. How
ever, my students face a lot of challenges. My students all live in poverty and in a dangerous neighbor
hood. Despite these challenges, I have students who have the the desire to defeat these challenges. My
students all have learning disabilities and currently all are performing below grade level. My students
are visual learners and will benefit from a classroom that fulfills their preferred learning style.The
materials I am requesting will allow my students to be prepared for the classroom with the necessary su
pplies.  Too often I am challenged with students who come to school unprepared for class due to economi
c challenges.  I want my students to be able to focus on learning and not how they will be able to get
school supplies.  The supplies will last all year.  Students will be able to complete written assignmen
ts and maintain a classroom journal.  The chart paper will be used to make learning more visual in clas
s and to create posters to aid students in their learning.  The students have access to a classroom pri
nter.  The toner will be used to print student work that is completed on the classroom Chromebooks.I wa
nt to try and remove all barriers for the students learning and create opportunities for learning. One
of the biggest barriers is the students not having the resources to get pens, paper, and folders. My st
udents 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 awhile, you could miss it.\"  from
the movie, Ferris Bueller's Day Off.  Think back...what do you remember about your grandparents?  How a
```

mazing 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 nonfiction books. Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson. They also love to read about insects, space and plants. My students are hungry bookworms! My students are eager to learn and read about the world around them. My kids love to be at school and are like little sponges absorbing everything around them. Their parents work long hours and usually do not see their children. My students are usually cared for by their grandparents or a family friend. Most of my students do not have someone who speaks English at home. Thus it is difficult for 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 and being able to share these memories with future generations will be a rewarding experience. As part of our social studies curriculum, students will 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 schools have changed over time. As a culminating activity, my students will capture a slice of their history and preserve it through scrap booking. Key important events in their young lives will be documented with the date, location, and names. Students will be using photos from home and from school to create their second grade memories. Their scrap books will preserve their unique stories for future generations to enjoy.Your donation to this project will provide my second graders with an opportunity to learn about social studies in a fun and creative manner. Through their scrapbooks, children will share their story with others and have a historical document for the rest of their lives.
==========

\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest 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\nStudents in my class come from a variety of different backgrounds which makes for wonderful sharing of experiences and cultures, including Native Americans.\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 materials and have many different opportunities to practice a skill before it is mastered. Having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the perfect place to learn about agriculture and nutrition. My students love to role play in our pretend kitchen 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 Cooking Lessons\" where we learn important math and writing concepts while cooking delicious healthy food for snack time. My students 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 it's healthy for their bodies. This project would expand our learning of nutrition 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 cookbooks to be printed and shared with families. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healthy cooking.nannan
==========

My classroom consists of twenty-two amazing sixth graders from different cultures and backgrounds. They are a social bunch who enjoy working in partners and working with groups. They are hard-working and eager to head to middle school next year. My job is to get them ready to make this transition 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 - choice on where to sit and work, the order to complete assignments, choice of 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 because we ALL share it together. Because my time with them is limited, I want to ensure they get the most of this time and enjoy it to the best of their 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 to choose their own spots while they are working independently or in groups. 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 making them stay at their desks! It is because of this that I am looking toward the “Flexible Seating” option for my classroom.\r\n The students look forward 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 options. I am requesting various seating so my students have more options to sit. Currently, I have a stool and a papasan chair I inherited from the previous sixth-grade teacher as well as five milk crate seats I made, but I would like to give them more options and reduce the competition for the “good seats”. I am also requesting two rugs as not only more seating options but to make the classroom more welcoming and appealing. In order for my students 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 not using them to leave more room for our flexible seating options.\r\nI know that with more seating options, they will be that much more excited about coming to school! Thank you for your support in making my classroom one students will remember forever!nannan
==========

In [14]:

```
# 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"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 [15]:

```
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 smallest students with the bigges
t enthusiasm for learning. My students learn in many different ways using all of our senses and multipl
e intelligences. I use a wide range of techniques to help all my students succeed. \r\nStudents in my c
lass come from a variety of different backgrounds which makes for wonderful sharing of experiences and
cultures, including Native Americans.\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. Kindergar
teners in my class love to work with hands-on materials and have many different opportunities to practi
ce a skill before it is mastered. Having the social skills to work cooperatively with friends is a cruc
ial aspect of the kindergarten curriculum.Montana is the perfect place to learn about agriculture and n
utrition. My students love to role play in our pretend kitchen 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 \"Com
mon Core Cooking Lessons\" where we learn important math and writing concepts while cooking delicious h
ealthy food for snack time. My students will have a grounded appreciation for the work that went into m
aking 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 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 cookbooks to be printed and shared with
families. \r\nStudents will gain math and literature skills as well as a life long enjoyment for health
y cooking.nannan
==================================================
```

In [16]:

```
# \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 smallest 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 wonderful sharing of experiences and cultu
res, 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 classroom. Kindergarteners
in my class love to work with hands-on materials and have many different opportunities to practice a sk
ill before it is mastered. Having the social skills to work cooperatively with friends is a crucial asp
ect of the kindergarten curriculum.Montana is the perfect place to learn about agriculture and nutritio
n. My students love to role play in our pretend kitchen in the early childhood classroom. I have had se
veral 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 math and writing concepts while cooking delicious healthy foo
d for snack time. My students 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 it is healthy for their bodies. Th
is project would expand our learning of nutrition and agricultural cooking recipes by having us peel ou
r own apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classr
oom garden in the spring. We will also create our own cookbooks to be printed and shared with families.
Students will gain math and literature skills as well as a life long enjoyment for healthy cooking.nann
an
```

In [17]:

```
#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 students with the biggest enthu
siasm for learning My students learn in many different ways using all of our senses and multiple intell
igences 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 wonderful sharing of experiences and cultures includ
ing 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 classroom Kindergarteners in my class lo
ve to work with hands on materials and have many different opportunities to practice a skill before it
is mastered Having the social skills to work cooperatively with friends is a crucial aspect of the kind
ergarten curriculum Montana is the perfect place to learn about agriculture and nutrition My students l
ove to role play in our pretend kitchen 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 Cooking Lessons where w
e learn important math and writing 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 whe
re the ingredients came from as well as how it is healthy for their bodies This project would expand ou
r learning of nutrition and agricultural cooking recipes by having us peel our own apples to make homem
ade applesauce make our own bread and mix up healthy plants from our classroom garden in the spring We
will also create our own cookbooks to be printed and shared with families Students will gain math and l
iterature skills as well as a life long enjoyment for healthy cooking nannan
```

In [18]:

```
# 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', 't
heir',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these',
'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'd
o', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'whil
e', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'bef
ore', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'a
gain', '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 [19]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar

# https://gist.github.com/sebleier/554280

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)
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [01:27<00:
00, 1249.54it/s]
```

In [20]:

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

Out[20]:

```
'person person no matter small dr seuss teach smallest students biggest enthusiasm learning students le
arn many different ways using senses multiple intelligences use wide range techniques help students suc
```

ceed students class come variety different backgrounds makes wonderful sharing experiences cultures inc
luding native americans school caring community successful learners seen collaborative student project
based learning classroom kindergarteners class love work hands materials many different opportunities p
ractice skill mastered social skills work cooperatively friends crucial aspect kindergarten curriculum
montana perfect place learn agriculture nutrition students love role play pretend kitchen early childho
od classroom several kids ask try cooking real food take idea create common core cooking lessons learn
important math writing concepts cooking delicious healthy food snack time students grounded appreciatio
n work went making food knowledge ingredients came well healthy bodies project would expand learning nu
trition agricultural cooking recipes us peel apples make homemade applesauce make bread mix healthy pla
nts classroom garden spring also create cookbooks printed shared families students gain math literature
skills well life long enjoyment healthy cooking nannan'

In [21]:

```
project_data['clean_essay'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
project_data.head(2)
```

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5

2.4 Preprocessing of `project_title`

In [22]:

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

In [23]:

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

```
Engineering STEAM into the Primary Classroom
==================================================
Building Blocks for Learning
==================================================
Empowering Students Through Art:Learning About Then and Now
==================================================
Health Nutritional Cooking in Kindergarten
==================================================
Turning to Flexible Seating: One Sixth-Grade Class's Journey to Freedom
==================================================
```

In [24]:

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

def decontracted(phrase):
    # specific
```

```
    # 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]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar

# https://gist.github.com/sebleier/554280

for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:03<00:0
0, 28599.41it/s]
```

In [26]:

```
# after preprocesing
preprocessed_titles[20000]
```

Out[26]:

```
'health nutritional cooking kindergarten'
```

In [27]:

```
project_data['clean_project_title'] = preprocessed_titles
project_data.drop(['project_title'], axis=1, inplace=True)
project_data.head(2)
```

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5

2.5 Cleaning data of project_grade_category

In [28]:

```
#cleaning project_grade_category
```

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
grade_list = []
for i in grades:
    i = i.replace('-','_')
    i = i.replace(' ','')

    grade_list.append(i)
```

In [29]:

```
project_data['clean_grade_category'] = grade_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)
```

Out[29]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_essay_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	I have been fortunate enough to use the Fairy ...
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Imagine being 8-years old. You're in your th...

2.6 Droping unnecessary columns

In [30]:

```
#project_data.drop(['id'], axis=1, inplace=True)
project_data.drop(['teacher_id'], axis=1, inplace=True)
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
project_data.drop(['project_resource_summary'], axis=1, inplace=True)
project_data.drop(['Unnamed: 0'], axis=1, inplace=True)
project_data.head(2)
```

Out[30]:

	id	teacher_prefix	school_state	Date	teacher_number_of_previously_posted_projects	project_is_a
55660	p205479	Mrs.	CA	2016-04-27 00:27:36	53	1
76127	p043609	Ms.	UT	2016-04-27 00:31:25	4	1

2.7 Adding price column in our dataframe

In [31]:

```
resource_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1541272 entries, 0 to 1541271
Data columns (total 4 columns):
id              1541272 non-null object
description     1540980 non-null object
quantity        1541272 non-null int64
price           1541272 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 47.0+ MB
```

In [32]:

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

Out[32]:

	id	teacher_prefix	school_state	Date	teacher_number_of_previously_posted_projects	project_is_a
55660	p205479	Mrs.	CA	2016-04-27 00:27:36	53	1
76127	p043609	Ms.	UT	2016-04-27 00:31:25	4	1

In [33]:

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

In [34]:

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

Out[34]:

	id	teacher_prefix	school_state	Date	teacher_number_of_previously_posted_projects	project_is_appro
0	p205479	Mrs.	CA	2016-04-27 00:27:36	53	1
1	p043609	Ms.	UT	2016-04-27 00:31:25	4	1

2.8 Preprocessing of teacher_prefix

In [35]:

```
import re
prefix = list(project_data['teacher_prefix'].values)

prefix_list = []
```

```
for i in prefix:

    j=str(i)
    j=j.lower()
    j = re.sub(r"\.", "",j)

    prefix_list.append(j)


#print(prefix_list)
```

In [36]:

```
project_data['clean_teacher_prefix'] = prefix_list
project_data.drop(['teacher_prefix'], axis=1, inplace=True)
project_data.head(2)
```

Out[36]:

	id	school_state	Date	teacher_number_of_previously_posted_projects	project_is_approved	clean_cate
0	p205479	CA	2016-04-27 00:27:36	53	1	Math_Scien
1	p043609	UT	2016-04-27 00:31:25	4	1	SpecialNeed

2.9 Preprocessing of school_state

In [37]:

```
state = list(project_data['school_state'].values)

state_list = []

for i in state:

    j=str(i)
    j=j.lower()


    state_list.append(j)


#print(state_list)
```

In [38]:

```
project_data['clean_school_state'] = state_list
#project_data.drop(['school_state'], axis=1, inplace=True)
project_data.head(2)
```

Out[38]:

	id	school_state	Date	teacher_number_of_previously_posted_projects	project_is_approved	clean_cate
0	p205479	CA	2016-04-27 00:27:36	53	1	Math_Scien
1	p043609	UT	2016-04-27	4	1	SpecialNeed

	id	school_state	Date	teacher_number_of_previously_posted_projects	project_is_approved	clean_cate

Assignment 3: Apply KNN

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

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
X, y = load_digits(return_X_y=True)
X.shape
X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
X_new.shape
========
output:
(1797, 64)
(1797, 20)
```

   - Repeat the steps 2 and 3 on the data matrix after feature selection
5. **Conclusion**
   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

Note: Data Leakage

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

3. K Nearest Neighbor

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

In [39]:

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


from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection


X = project_data.drop(['project_is_approved','id'], axis=1)
X.head(2)

y = project_data['project_is_approved'].values


# split the data set into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,shuffle=False)

# split the train data set into cross validation train and cross validation test
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.2,shuffle=False)

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

```
(69918, 11) (69918,)
(17480, 11) (17480,)
(21850, 11) (21850,)
```

In [40]:

```
X.head(2)
```

Out[40]:

	school_state	Date	teacher_number_of_previously_posted_projects	clean_categories	clean_subcategories	cl
0	CA	2016-04-27 00:27:36	53	Math_Science	AppliedSciences Health_LifeScience	fo e fa st cl
1	UT	2016-04-27 00:31:25	4	SpecialNeeds	SpecialNeeds	in y th cl s

3.2 Make Data Model Ready: encoding numerical, categorical features

In [ ]:

```
# please write all the code with proper documentation, and proper titles for each subsection
```

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

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

3.2.1 encoding categorical features: School State

In [41]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_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['clean_school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['clean_school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['clean_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
(69918, 51) (69918,)
(17480, 51) (17480,)
(21850, 51) (21850,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks',
'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', '
ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
====================================================================================================
```

3.2.2 encoding categorical features: teacher_prefix

In [42]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_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['clean_teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer.transform(X_cv['clean_teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['clean_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
(69918, 6) (69918,)
(17480, 6) (17480,)
(21850, 6) (21850,)
['dr', 'mr', 'mrs', 'ms', 'nan', 'teacher']
====================================================================================================
```

3.2.3 encoding categorical features: project_grade_category

In [43]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_grade_category'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
```

```
X_train_grade_ohe = vectorizer.transform(X_train['clean_grade_category'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['clean_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['clean_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
(69918, 4) (69918,)
(17480, 4) (17480,)
(21850, 4) (21850,)
['grades3_5', 'grades6_8', 'grades9_12', 'gradesprek_2']
====================================================================================================
```

3.2.4 encoding categorical features: project_subject_categories

In [44]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data

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

print("After vectorizations")
print(X_train_categories_ohe.shape, y_train.shape)
print(X_cv_categories_ohe.shape, y_cv.shape)
print(X_test_categories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(69918, 9) (69918,)
(17480, 9) (17480,)
(21850, 9) (21850,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_scienc
e', 'music_arts', 'specialneeds', 'warmth']
====================================================================================================
```

3.2.5 encoding categorical features: project_subject_subcategories

In [45]:

```
vectorizer = CountVectorizer()
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_subcategories_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_cv_subcategories_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_subcategories_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_subcategories_ohe.shape, y_train.shape)
print(X_cv_subcategories_ohe.shape, y_cv.shape)
print(X_test_subcategories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(69918, 30) (69918,)
(17480, 30) (17480,)
(21850, 30) (21850,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_careerprep', 'co
mmunityservice', 'earlydevelopment', 'economics', 'environmentalscience', 'esl', 'extracurricular', 'fi
nancialliteracy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wellness', 'history_
geography', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutritioneducation', 'other', 'p
arentinvolvement', 'performingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'war
mth']
====================================================================================================
```

3.2.6 encoding numerical feature: price

In [46]:

```
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))


X_train_price_norm = X_train_price_norm.reshape(-1,1)
X_cv_price_norm = X_cv_price_norm.reshape(-1,1)
X_test_price_norm = X_test_price_norm.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
(69918, 1) (69918,)
(17480, 1) (17480,)
(21850, 1) (21850,)
====================================================================================================
```

3.2.7 encoding numerical feature: teacher_number_of_previously_posted_projects

In [47]:

```
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_posted_project_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_project
s'].values.reshape(1,-1))
X_cv_posted_project_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].va
lues.reshape(1,-1))
X_test_posted_project_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'
].values.reshape(1,-1))

X_train_posted_project_norm = X_train_posted_project_norm.reshape(-1,1)
X_cv_posted_project_norm = X_cv_posted_project_norm.reshape(-1,1)
X_test_posted_project_norm = X_test_posted_project_norm.reshape(-1,1)


print("After vectorizations")
print(X_train_posted_project_norm.shape, y_train.shape)
print(X_cv_posted_project_norm.shape, y_cv.shape)
print(X_test_posted_project_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(69918, 1) (69918,)
(17480, 1) (17480,)
(21850, 1) (21850,)
====================================================================================================
```

3.3 Make Data Model Ready: encoding eassay, and project_title

In [48]:

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

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

3.3.1 encoding essay

3.3.1.1 encoding essay : BOW

In [49]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(project_data['clean_essay'].values)

X_train_essay_bow = vectorizer.transform(X_train['clean_essay'].values)
X_cv_essay_bow = vectorizer.transform(X_cv['clean_essay'].values)
X_test_essay_bow = vectorizer.transform(X_test['clean_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(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(69918, 16512) (69918,)
(17480, 16512) (17480,)
(21850, 16512) (21850,)
====================================================================================================
```

3.3.1.2 encoding essay : TFIDF

In [50]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(project_data['clean_essay'].values)

X_train_essay_tfidf = vectorizer.transform(X_train['clean_essay'].values)
X_cv_essay_tfidf= vectorizer.transform(X_cv['clean_essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['clean_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(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(69918, 16512) (69918,)
(17480, 16512) (17480,)
(21850, 16512) (21850,)
====================================================================================================
```

3.3.1.3 encoding essay : AVG W2V

In [51]:

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

In [52]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essay_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_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_essay_train.append(vector)

print(len(avg_w2v_essay_train))
print(len(avg_w2v_essay_train[0]))
#print(avg_w2v_essay_train[0])
```

```
100%|| 69918/69918 [00:30<00:00, 2262.40it/s]
```

```
69918
300
```

In [53]:

```
avg_w2v_essay_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_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_essay_cv.append(vector)
```

```
100%|| 17480/17480 [00:07<00:00, 2467.50it/s]
```

In [54]:

```
avg_w2v_essay_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_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_essay_test.append(vector)
```

```
100%|| 21850/21850 [00:08<00:00, 2491.31it/s]
```

3.3.1.4 encoding essay : TFIDF W2V

In [55]:

```
# Similarly you can vectorize for essay

tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['clean_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 [56]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
essay_tfidf_w2v_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_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((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_train.append(vector)

print(len(essay_tfidf_w2v_train))
print(len(essay_tfidf_w2v_train[0]))
```

```
100%|██████████| 69918/69918 [03:05<00
:00, 376.25it/s]
```

```
69918
300
```

In [57]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
essay_tfidf_w2v_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_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((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_cv.append(vector)

print(len(essay_tfidf_w2v_cv))
print(len(essay_tfidf_w2v_cv[0]))
```

```
100%|██████████| 17480/17480 [00:44<00
:00, 391.64it/s]
```

```
17480
300
```

In [58]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
essay_tfidf_w2v_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_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((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
```

```
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_test.append(vector)

print(len(essay_tfidf_w2v_test))
print(len(essay_tfidf_w2v_test[0]))
```

```
100%|██████████| 21850/21850 [00:55<00
:00, 395.47it/s]
```

```
21850
300
```

3.3.2 encoding titles

3.3.2.1 encoding titles : BOW

In [59]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(project_data['clean_project_title'].values)

X_train_title_bow = vectorizer.transform(X_train['clean_project_title'].values)
X_cv_title_bow = vectorizer.transform(X_cv['clean_project_title'].values)
X_test_title_bow = vectorizer.transform(X_test['clean_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(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(69918, 3222) (69918,)
(17480, 3222) (17480,)
(21850, 3222) (21850,)
====================================================================================================
```

3.3.2.2 encoding titles : TFIDF

In [60]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(project_data['clean_project_title'].values)

X_train_title_tfidf = vectorizer.transform(X_train['clean_project_title'].values)
X_cv_title_tfidf= vectorizer.transform(X_cv['clean_project_title'].values)
X_test_title_tfidf = vectorizer.transform(X_test['clean_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(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(69918, 3222) (69918,)
(17480, 3222) (17480,)
(21850, 3222) (21850,)
====================================================================================================
```

3.3.2.3 encoding titles : AVG W2V

In [61]:

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

```
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [62]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_title_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_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_title_train.append(vector)

print(len(avg_w2v_title_train))
print(len(avg_w2v_title_train[0]))
#print(avg_w2v_title_train[0])
```

```
100%|██████████| 69918/69918 [00:01<00:0
0, 54564.62it/s]
```

```
69918
300
```

In [63]:

```
avg_w2v_title_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_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_title_cv.append(vector)
```

```
100%|██████████| 17480/17480 [00:00<00:0
0, 48647.36it/s]
```

In [64]:

```
avg_w2v_title_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_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_title_test.append(vector)
```

```
100%|██████████| 21850/21850 [00:00<00:0
0, 53718.57it/s]
```

3.3.2.4 encoding titles : TFIDF W2V

In [65]:

```
# Similarly you can vectorize for title also

tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['clean_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 [66]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
title_tfidf_w2v_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_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((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_train.append(vector)

print(len(title_tfidf_w2v_train))
print(len(title_tfidf_w2v_train[0]))
```

```
100%|██████████| 69918/69918 [00:02<00:0
0, 23804.41it/s]
```

```
69918
300
```

In [67]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
title_tfidf_w2v_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_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((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_cv.append(vector)

print(len(title_tfidf_w2v_cv))
print(len(title_tfidf_w2v_cv[0]))
```

```
100%|██████████| 17480/17480 [00:00<00:0
0, 23790.56it/s]
```

```
17480
300
```

In [68]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
title_tfidf_w2v_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_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((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
```

```
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_test.append(vector)

print(len(title_tfidf_w2v_test))
print(len(title_tfidf_w2v_test[0]))
```

```
100%|██████████| 21850/21850 [00:01<00:00, 21198.45it/s]
```

```
21850
300
```

3.4 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 [ ]:

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

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

3.4.1 Applying KNN brute force on BOW, SET 1

In [69]:

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

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

X_tr_bow = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_ohe, X
_train_subcategories_ohe, X_train_price_norm, X_train_posted_project_norm, X_train_essay_bow, X_train_t
itle_bow)).tocsr()
X_cr_bow = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_categories_ohe, X_cv_subcateg
ories_ohe, X_cv_price_norm, X_cv_posted_project_norm, X_cv_essay_bow, X_cv_title_bow)).tocsr()
X_te_bow = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe, X_tes
t_subcategories_ohe, X_test_price_norm, X_test_posted_project_norm, X_test_essay_bow, X_test_title_bow)
).tocsr()

y_train_bow = y_train
y_cv_bow = y_cv
y_test_bow = y_test

print("Final Data matrix")
print(X_tr_bow.shape, y_train_bow.shape)
print(X_cr_bow.shape, y_cv_bow.shape)
print(X_te_bow.shape, y_test_bow.shape)
print("="*100)
```

```
Final Data matrix
(69918, 19836) (69918,)
(17480, 19836) (17480,)
(21850, 19836) (21850,)
====================================================================================================
```

3.4.1.1 Hyper parameter tuning

In [70]:

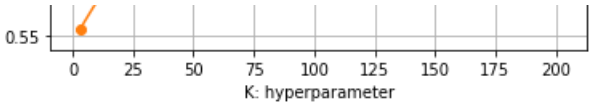
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
```

```
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 [72]:

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

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thr
esholded 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 = [3, 15, 25, 51, 101,151,201]
for k in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=k, n_jobs=-1)
    neigh.fit(X_tr_bow, y_train_bow)

    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_bow,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv_bow, 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 [58:45<00
:00, 504.31s/it]
```



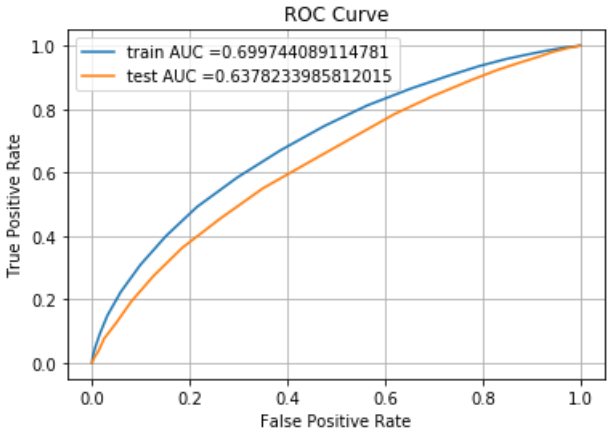
3.4.1.2 Testing the performance of the model on test data, plotting ROC Curves

In [146]:

```
best_k = 51
```

In [147]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_tr_bow, y_train_bow)
# 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_bow, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_bow, 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("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.grid()
plt.show()
```



In [75]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [76]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
# print("Train confusion matrix")
# print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
# print("Test confusion matrix")
# print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.4115089378734088 for threshold 0.784
```

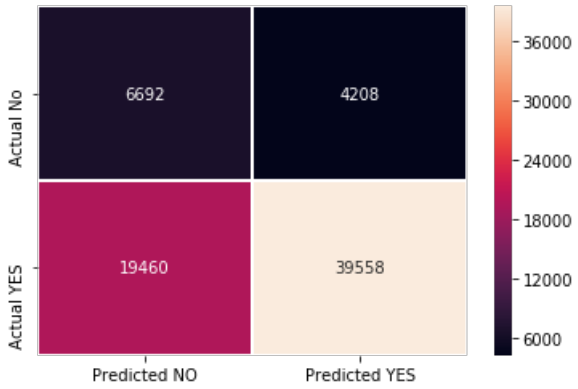
In [77]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [78]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
```

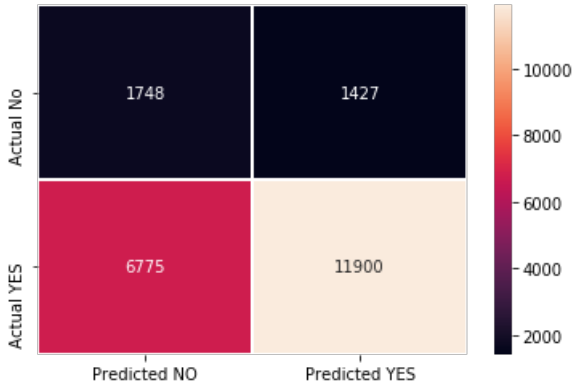
```
Train confusion matrix
```



In [79]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



3.4.2 Applying KNN brute force on TFIDF, SET 2

In [80]:

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

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

```
X_tr_tfidf = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_ohe,
X_train_subcategories_ohe, X_train_price_norm, X_train_posted_project_norm, X_train_essay_tfidf, X_trai
n_title_tfidf)).tocsr()
X_cr_tfidf = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_categories_ohe, X_cv_subcat
egories_ohe, X_cv_price_norm, X_cv_posted_project_norm, X_cv_essay_tfidf, X_cv_title_tfidf)).tocsr()
X_te_tfidf = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe, X_t
est_subcategories_ohe, X_test_price_norm, X_test_posted_project_norm, X_test_essay_tfidf, X_test_title_
tfidf)).tocsr()

y_train_tfidf = y_train
y_cv_tfidf = y_cv
y_test_tfidf = y_test

print("Final Data matrix")
print(X_tr_tfidf.shape, y_train_tfidf.shape)
print(X_cr_tfidf.shape, y_cv_tfidf.shape)
print(X_te_tfidf.shape, y_test_tfidf.shape)
print("="*100)
```

```
Final Data matrix
(69918, 19836) (69918,)
(17480, 19836) (17480,)
(21850, 19836) (21850,)
====================================================================================================
```

3.4.2.1 Hyper parameter tuning

In [81]:

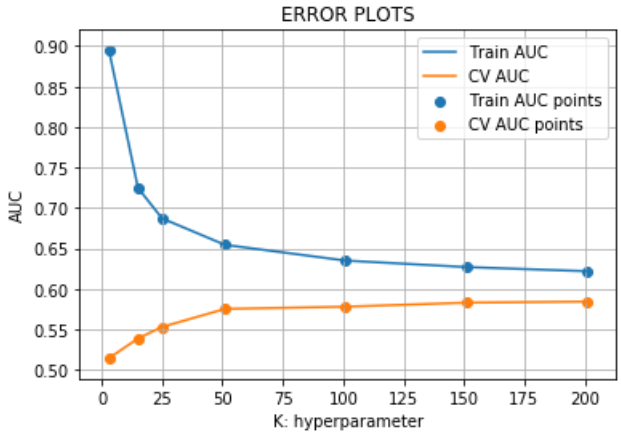
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
    # not the predicted outputs

    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 [82]:

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

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thr
esholded 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 = [3, 15, 25, 51, 101,151,201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_tfidf, y_train_tfidf)

    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_tfidf,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv_tfidf, 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 [1:03:35<00
:00, 542.86s/it]
```



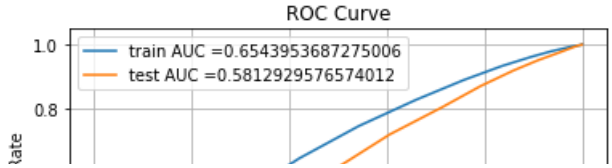
3.4.2.2 Testing the performance of the model on test data, plotting ROC Curves

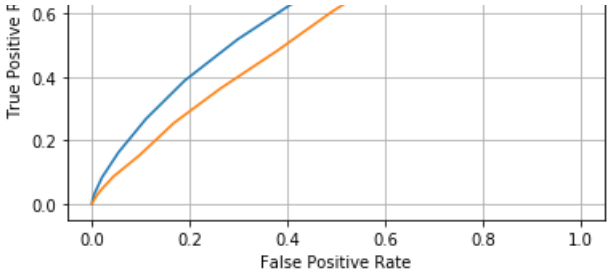
In [148]:

```
best_k=51
```

In [149]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_tr_tfidf, y_train_tfidf)
# 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_tfidf, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidf, 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("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.grid()
plt.show()
```





In [85]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [86]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.3721172455239126 for threshold 0.843
```

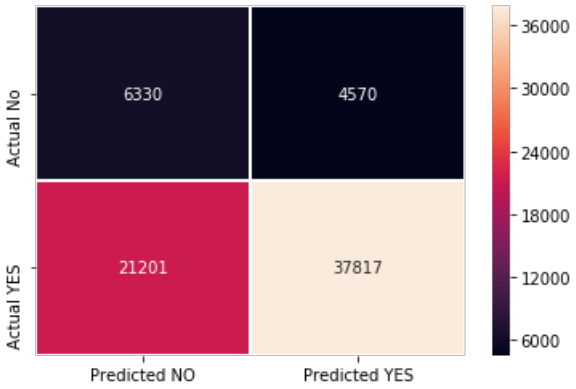
In [87]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [88]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
```

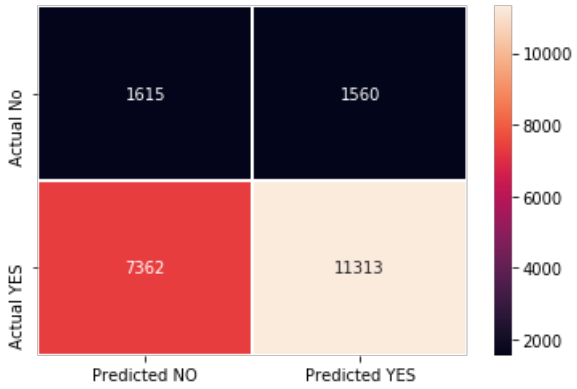
```
Train confusion matrix
```



In [89]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



3.4.3 Applying KNN brute force on AVG W2V, SET 3

In [90]:

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

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

X_tr_avgw2v = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_ohe
, X_train_subcategories_ohe, X_train_price_norm, X_train_posted_project_norm, avg_w2v_essay_train, avg_
w2v_title_train)).tocsr()
X_cr_avgw2v = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_categories_ohe, X_cv_subca
tegories_ohe, X_cv_price_norm, X_cv_posted_project_norm, avg_w2v_essay_cv, avg_w2v_title_cv)).tocsr()
X_te_avgw2v = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe, X_
test_subcategories_ohe, X_test_price_norm, X_test_posted_project_norm, avg_w2v_essay_test, avg_w2v_titl
e_test)).tocsr()

y_train_avgw2v = y_train
y_cv_avgw2v = y_cv
y_test_avgw2v = y_test

print("Final Data matrix")
print(X_tr_avgw2v.shape, y_train_avgw2v.shape)
print(X_cr_avgw2v.shape, y_cv_avgw2v.shape)
print(X_te_avgw2v.shape, y_test_avgw2v.shape)
print("="*100)
```

```
Final Data matrix
(69918, 702) (69918,)
(17480, 702) (17480,)
(21850, 702) (21850,)
====================================================================================================
```

In [91]:

```
#taking 50k data point which is almost half of each train, test and cv dataset

tr_index=int(X_tr_avgw2v.get_shape()[0]/2)
X_tr_avgw2v = X_tr_avgw2v[:tr_index:]
y_tr_avgw2v = y_train_avgw2v[:tr_index:]
print(X_tr_avgw2v.shape,y_tr_avgw2v.shape)

crv_index=int(X_cr_avgw2v.get_shape()[0]/2)
X_crv_avgw2v = X_cr_avgw2v[:crv_index:]
y_crv_avgw2v = y_cv_avgw2v[:crv_index:]
print(X_crv_avgw2v.shape,y_crv_avgw2v.shape)

test_index=int(X_te_avgw2v.get_shape()[0]/2)
X_ts_avgw2v = X_te_avgw2v[:test_index:]
y_ts_avgw2v = y_test_avgw2v[:test_index:]
print(X_ts_avgw2v.shape,y_ts_avgw2v.shape)
```

```
(34959, 702) (34959,)
(8740, 702) (8740,)
(10925, 702) (10925,)
```

3.4.3.1 Hyperparameter tuning

In [92]:

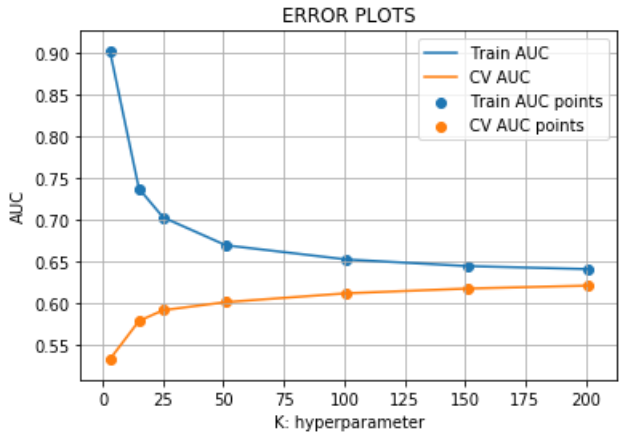
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
    # not the predicted outputs

    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 [94]:

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

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thr
esholded 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 = [3, 15, 25, 51, 101,151,201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_avgw2v, y_tr_avgw2v)

    y_train_pred = batch_predict(neigh, X_tr_avgw2v)
    y_cv_pred = batch_predict(neigh, X_crv_avgw2v)

    # 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_tr_avgw2v,y_train_pred))
    cv_auc.append(roc_auc_score(y_crv_avgw2v, 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()
```

```
  0%|
| 0/7 [00:00<?, ?it/s]
 14%|
| 1/7 [20:15<2:01:35, 1215.85s/it]
 29%|
| 2/7 [44:02<1:46:35, 1279.03s/it]
 43%|
| 3/7 [1:04:16<1:23:58, 1259.62s/it]
 57%|
| 4/7 [1:24:53<1:02:38, 1252.73s/it]
 71%|
| 5/7 [1:44:56<41:15, 1237.93s/it]
```

```
 86%|                                                                                 | 6/7 [2:04:52<20:25, 1225.43s/it]
100%|                                                                                | 7/7 [2:24:57<00:00, 1219.26s/it]
```



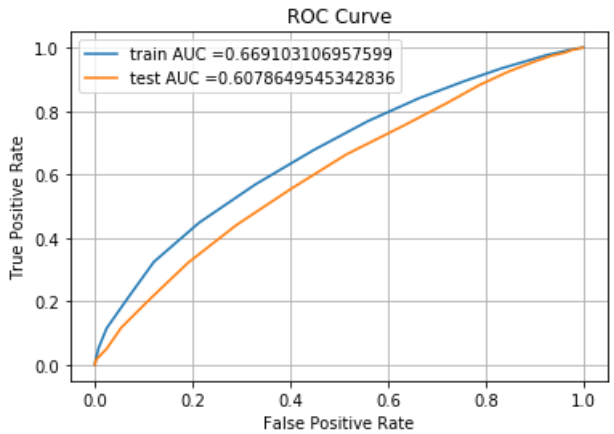
3.4.3.2 Testing the performance of the model on test data, plotting ROC Curves

In [150]:

```
best_k = 51
```

In [151]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_tr_avgw2v, y_tr_avgw2v)
# 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_avgw2v)
y_test_pred = batch_predict(neigh, X_ts_avgw2v)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_tr_avgw2v, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_ts_avgw2v, 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("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.grid()
plt.show()
```



In [98]:

```
```

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [99]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.3821241369474735 for threshold 0.863
```

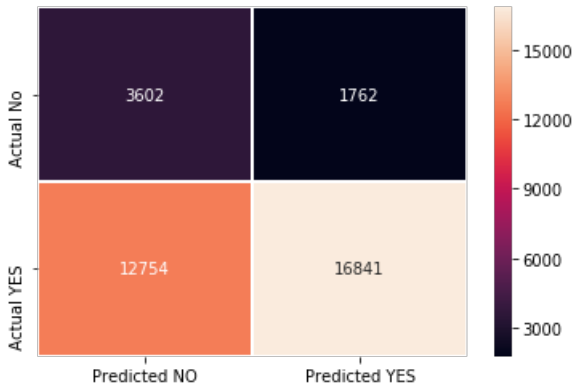
In [100]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [102]:

```
print("Train confusion matrix")
get_confusion_matrix(y_tr_avgw2v, predict_with_best_t(y_train_pred, best_t))
```

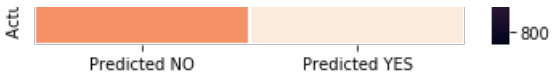
```
Train confusion matrix
```



In [103]:

```
print("Test confusion matrix")
get_confusion_matrix(y_ts_avgw2v, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



3.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [104]:

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

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

X_tr_tfidfw2v = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_o
he, X_train_subcategories_ohe, X_train_price_norm, X_train_posted_project_norm, essay_tfidf_w2v_train,
title_tfidf_w2v_train)).tocsr()
X_cr_tfidfw2v = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_categories_ohe, X_cv_sub
categories_ohe, X_cv_price_norm, X_cv_posted_project_norm, essay_tfidf_w2v_cv, title_tfidf_w2v_cv)).toc
sr()
X_te_tfidfw2v = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe,
X_test_subcategories_ohe, X_test_price_norm, X_test_posted_project_norm, essay_tfidf_w2v_test, title_tf
idf_w2v_test)).tocsr()

y_train_tfidfw2v = y_train
y_cv_tfidfw2v = y_cv
y_test_tfidfw2v = y_test

print("Final Data matrix")
print(X_tr_tfidfw2v.shape, y_train_tfidfw2v.shape)
print(X_cr_tfidfw2v.shape, y_cv_tfidfw2v.shape)
print(X_te_tfidfw2v.shape, y_test_tfidfw2v.shape)
print("="*100)
```

```
Final Data matrix
(69918, 702) (69918,)
(17480, 702) (17480,)
(21850, 702) (21850,)
====================================================================================================
```

In [105]:

```
#taking 50k data point which is almost half of each train, test and cv dataset

tr_index=int(X_tr_tfidfw2v.get_shape()[0]/2)
X_tr_tfidfw2v = X_tr_tfidfw2v[:tr_index:]
y_tr_tfidfw2v = y_train_tfidfw2v[:tr_index:]
print(X_tr_tfidfw2v.shape,y_tr_tfidfw2v.shape)

crv_index=int(X_cr_tfidfw2v.get_shape()[0]/2)
X_crv_tfidfw2v = X_cr_tfidfw2v[:crv_index:]
y_crv_tfidfw2v = y_cv_tfidfw2v[:crv_index:]
print(X_crv_tfidfw2v.shape,y_crv_tfidfw2v.shape)

test_index=int(X_te_tfidfw2v.get_shape()[0]/2)
X_ts_tfidfw2v = X_te_tfidfw2v[:test_index:]
y_ts_tfidfw2v = y_test_tfidfw2v[:test_index:]
print(X_ts_tfidfw2v.shape,y_ts_tfidfw2v.shape)
```

```
(34959, 702) (34959,)
(8740, 702) (8740,)
(10925, 702) (10925,)
```

3.4.4.1 Hyperparameter tuning

In [106]:

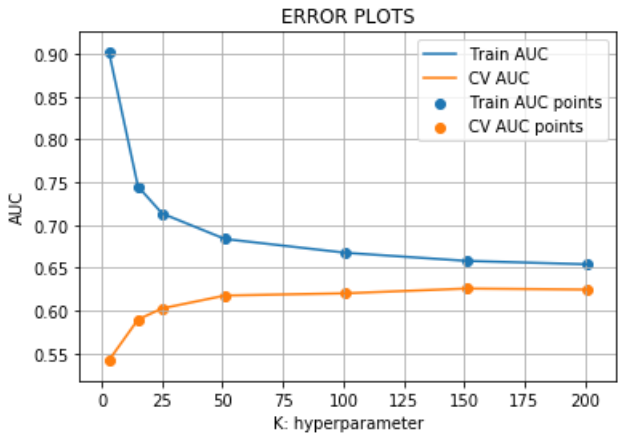
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
    # not the predicted outputs

    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 [125]:

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

train_auc = []
cv_auc = []
K = [3, 15, 25, 51, 101,151,201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_tfidfw2v, y_tr_tfidfw2v)

    y_train_pred = batch_predict(neigh, X_tr_tfidfw2v)
    y_cv_pred = batch_predict(neigh, X_crv_tfidfw2v)

    # 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_tr_tfidfw2v,y_train_pred))
    cv_auc.append(roc_auc_score(y_crv_tfidfw2v, 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()
```

```
  0%|                                                                                                 | 0/7
[00:00<?, ?it/s]
 14%|[redacted]                                                                                       | 1/7 [22:36<2:15:
41, 1356.99s/it]
 29%|[redacted]                                                                                       | 2/7 [45:19<1:53:
12, 1358.60s/it]
 43%|[redacted]                                                                                       | 3/7 [1:08:02<1:30:
40, 1360.01s/it]
 57%|[redacted]                                                                                       | 4/7 [1:31:27<1:08:
40, 1373.54s/it]
 71%|[redacted]                                                                                       | 5/7 [1:55:16<46:
20, 1390.18s/it]
 86%|[redacted]                                                                                       | 6/7 [2:20:23<23:
45, 1425.08s/it]
100%|[redacted]| 7/7 [2:45:43<00:
00, 1453.70s/it]
```



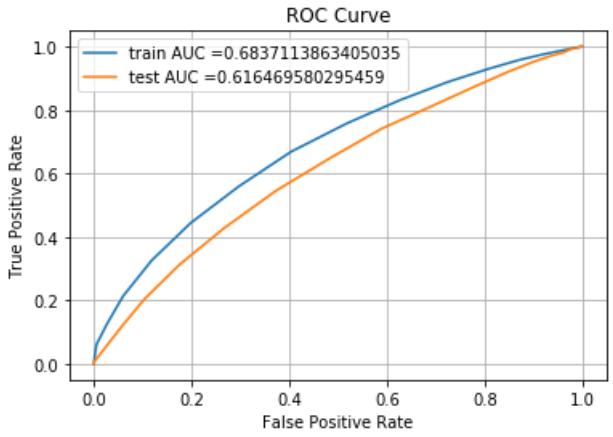
3.4.4.2 Testing the performance of the model on test data, plotting ROC Curves

In [152]:

```
best_k = 51
```

In [153]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_tr_tfidfw2v, y_tr_tfidfw2v)
# 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_tfidfw2v)
y_test_pred = batch_predict(neigh, X_ts_tfidfw2v)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_tr_tfidfw2v, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_ts_tfidfw2v, 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("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.grid()
plt.show()
```



In [109]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [110]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
==========================================================================================
the maximum value of tpr*(1-fpr) 0.3989629322223369 for threshold 0.843
```

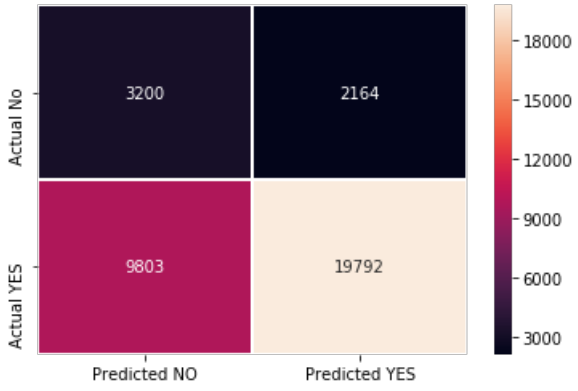
In [111]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [112]:

```
print("Train confusion matrix")
get_confusion_matrix(y_tr_tfidfw2v, predict_with_best_t(y_train_pred, best_t))
```

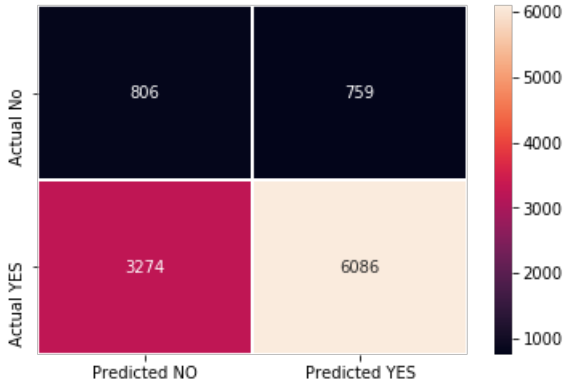
```
Train confusion matrix
```



In [113]:

```
print("Test confusion matrix")
get_confusion_matrix(y_ts_tfidfw2v, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



3.5 Feature selection with `SelectKBest`

In [111]:

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

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

In [76]:

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

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


# X_tr = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_ohe, X_train_subcategories_ohe, X_train_price_norm, X_train_posted_project_norm, X_train_essay_tfidf, X_train_title_tfidf)).tocsr()
# X_cr = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_categories_ohe, X_cv_subcategories_ohe, X_cv_price_norm, X_cv_posted_project_norm, X_cv_essay_tfidf, X_cv_title_tfidf)).tocsr()
# X_te = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe, X_test_subcategories_ohe, X_test_price_norm, X_test_posted_project_norm, X_test_essay_tfidf, X_test_title_tfidf)).tocsr()

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

```
Final Data matrix
(69918, 19836) (69918,)
(17480, 19836) (17480,)
(21850, 19836) (21850,)
====================================================================================================
```

In [77]:

```
# #combining X_tr X_cr, X_te for tfidf

# from scipy.sparse import vstack
# X = vstack((X_tr,X_cr,X_te))
# X.shape
```

Out[77]:

```
(109248, 19836)
```

In [78]:

```
# #similarly combining y for tfidf

# y = np.concatenate((y_train,y_cv,y_test))
# y.shape
```

Out[78]:

```
(109248,)
```

In [79]:

```
# print(y)
```

```
[1 1 1 ... 1 1 1]
```

In [ ]:

```
# print("Final Data matrix")
# print(X_tr_tfidf.shape, y_train_tfidf.shape)
# print(X_cr_tfidf.shape, y_cv_tfidf.shape)
# print(X_te_tfidf.shape, y_test_tfidf.shape)
# print("="*100)
```

In [134]:

```
# selecting top 2000 features

from sklearn.feature_selection import SelectKBest, f_classif

selector  = SelectKBest( f_classif, k=2000)
selector.fit(X_tr_tfidf, y_train_tfidf)


print(X_tr_tfidf.shape)
X_tr_new = selector.transform(X_tr_tfidf)
```

```
print(X_tr_new.shape,"\n")

print(X_cr_tfidf.shape)
X_cr_new = selector.transform(X_cr_tfidf)
print(X_cr_new.shape,"\n")

print(X_te_tfidf.shape)
X_te_new = selector.transform(X_te_tfidf)
print(X_te_new.shape,"\n")
```

```
(69918, 19836)
(69918, 2000)

(17480, 19836)
(17480, 2000)

(21850, 19836)
(21850, 2000)
```

In [81]:

```
# # split the data set into train and test after selecting 2000 features

# X_tr_fs, X_test_fs, y_tr_fs, y_test_fs = train_test_split(X_new, y, test_size=0.2,shuffle=False)

# # split the train data set into cross validation train and cross validation test
# X_tr_fs, X_cv_fs, y_tr_fs, y_cv_fs = train_test_split(X_tr_fs, y_tr_fs, test_size=0.2,shuffle=False)

# print(X_tr_fs.shape, y_tr_fs.shape)
# print(X_cv_fs.shape, y_cv_fs.shape)
# print(X_test_fs.shape, y_test_fs.shape)
```

```
(69918, 2000) (69918,)
(17480, 2000) (17480,)
(21850, 2000) (21850,)
```

3.5.1 Hyperparameter Tuning

In [135]:

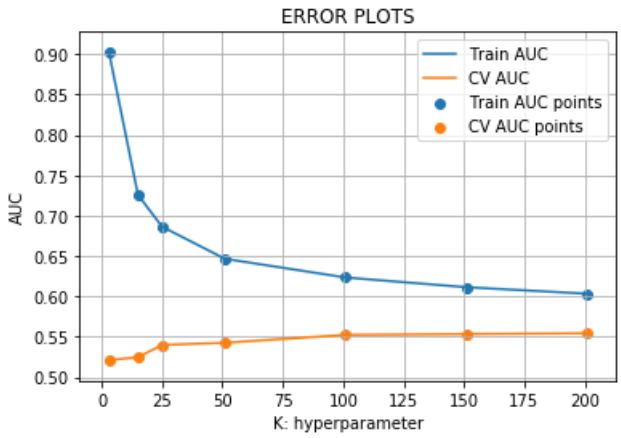
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
    # not the predicted outputs

    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 [136]:

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

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thr
esholded 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 = [3, 15, 25, 51, 101,151,201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_tr_new, y_train_tfidf)

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

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

```
  0%|
| 0/7 [00:00<?, ?it/s]
 14%|
| 1/7 [06:20<38:05, 380.96s/it]
 29%|
| 2/7 [13:25<32:49, 393.90s/it]
 43%|
| 3/7 [20:29<26:51, 402.94s/it]
 57%|
| 4/7 [27:33<20:28, 409.49s/it]
 71%|
| 5/7 [34:38<13:47, 413.95s/it]
 86%|
                    | 6/7 [41:41<06:56, 416.88s/it]
100%|
                                    | 7/7 [48:58<00:00, 422.72s/it]
```

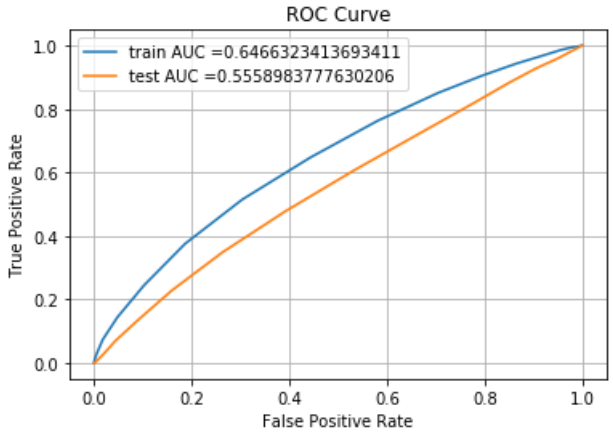


3.5.2 Testing the performance of the model on test data, plotting ROC Curves

In [140]:

```
# 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=51, n_jobs=-1)
neigh.fit(X_tr_new, y_train_tfidf)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive clas
s
# not the predicted outputs
```

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

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_tfidf, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidf, 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("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.grid()
plt.show()
```



In [141]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [142]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.360601495858362 for threshold 0.843
```

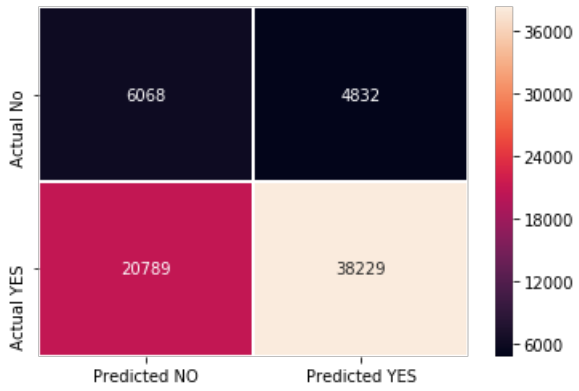
In [143]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [144]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train_tfidf, predict_with_best_t(y_train_pred, best_t))
```

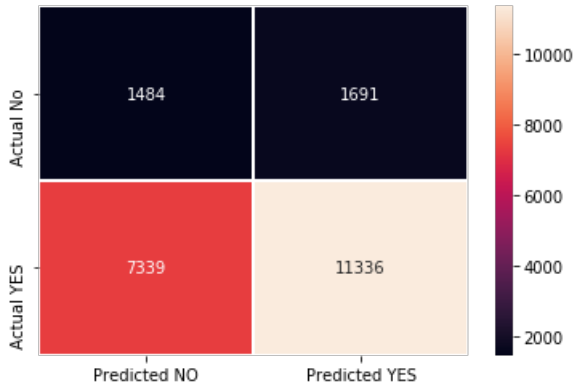
```
Train confusion matrix
```



In [145]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test_tfidfw2v, predict_with_best_t(y_test_pred, best_t))
```

Test confusion matrix



4. Conclusions

In [129]:

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

from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperparameter", "AUC"]

x.add_row(["BOW", "Brute", 51, 0.6378])
x.add_row(["TFIDF", "Brute", 51, 0.5812])
x.add_row(["ACG W2V", "Brute", 51, 0.6078])
x.add_row(["TFIDF W2V", "Brute", 51, 0.6174])

print(x)
```

```
+------------+-------+----------------+--------+
| Vectorizer | Model | Hyperparameter |  AUC   |
+------------+-------+----------------+--------+
|    BOW     | Brute |       51       | 0.6378 |
|   TFIDF    | Brute |       51       | 0.5812 |
|  ACG W2V   | Brute |       51       | 0.6078 |
| TFIDF W2V  | Brute |       51       | 0.6174 |
+------------+-------+----------------+--------+
```

TFIDF AUC is 0.5812 with 19836 features. After feature selection (2000 features) we got AUC of 0.5558