1. **KNN has been applied to the following 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)

    - **Hyper paramter tuning was done to find best K**

Description of the Dataset

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`

Feature	Description
	• Warmth **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Label	Description

Notes on the Essay Data

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

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

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

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

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

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import gensim
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
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

```
import csv
with open('resources.csv', 'r') as inp, open('first_edit.csv', 'w') as out:
    writer = csv.writer(out)
    for row in csv.reader(inp):
        if len(row) == 4:
            writer.writerow(row)
```

```
            writer.writerow(row)
```

In [3]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('first_edit.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]:

```
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 (4577891, 4)
['id' 'description' 'quantity' 'price']
```

Out[5]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [6]:

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

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

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

from collections import Counter
```

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

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

1.3 preprocessing of `project_subject_subcategories`

In [7]:

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

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

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

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

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

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

1.3 Text preprocessing

In [8]:

```
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [9]:

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

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [10]:

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

In [11]:

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

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==================================================
```

```
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade.  This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch.  Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students are on free or reduced lunch. We aren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character.In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use.  The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [12]:

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

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

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

In [13]:

```
sent = decontracted(project_data['essay'].values[20000])
```

```
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students receive free or reduced pr
ice lunch.  Despite their disabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able
to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids do not want to sit and do worksheets. They want to learn to count by j
umping and playing. Physical engagement is the key to our success. The number toss and color and s
hape mats can make that happen. My students will forget they are doing work and just have the fun
a 6 year old deserves.nannan
==================================================
```

In [14]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations.     The materials we have are the ones I seek out for
my students. I teach in a Title I school where most of the students receive free or reduced price
lunch.  Despite their disabilities and limitations, my students love coming to school and come eag
er to learn and explore.Have you ever felt like you had ants in your pants and you needed to groov
e and move as you were in a meeting? This is how my kids feel all the time. The want to be able to
move as they learn or so they say.Wobble chairs are the answer and I love then because they develo
p their core, which enhances gross motor and in Turn fine motor skills.   They also want to learn t
hrough games, my kids do not want to sit and do worksheets. They want to learn to count by jumping
and playing. Physical engagement is the key to our success. The number toss and color and shape ma
ts can make that happen. My students will forget they are doing work and just have the fun a 6 yea
r old deserves.nannan
```

In [15]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitiv
e delays gross fine motor delays to autism They are eager beavers and always strive to work their
hardest working past their limitations The materials we have are the ones I seek out for my studen
ts I teach in a Title I school where most of the students receive free or reduced price lunch
Despite their disabilities and limitations my students love coming to school and come eager to lea
rn and explore Have you ever felt like you had ants in your pants and you needed to groove and mov
e as you were in a meeting This is how my kids feel all the time The want to be able to move as th
ey learn or so they say Wobble chairs are the answer and I love then because they develop their co
re which enhances gross motor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Ph
ysical engagement is the key to our success The number toss and color and shape mats can make that
happen My students will forget they are doing work and just have the fun a 6 year old deserves nan
nan
```

In [16]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
```

```
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [17]:

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

```
100%|██████████| 109248/109248 [01:12<00:00, 1504.48it/s]
```

In [18]:

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

Out[18]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gros
s fine motor delays autism they eager beavers always strive work hardest working past limitations
the materials ones i seek students i teach title i school students receive free reduced price lunc
h despite disabilities limitations students love coming school come eager learn explore have ever
felt like ants pants needed groove move meeting this kids feel time the want able move learn say w
obble chairs answer i love develop core enhances gross motor turn fine motor skills they also want
learn games kids not want sit worksheets they want learn count jumping playing physical engagement
key success the number toss color shape mats make happen my students forget work fun 6 year old de
serves nannan'
```

1.4 Preprocessing of `project_title`

In [19]:

```
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
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)
    # https://gist.github.com/sebleier/554280
    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<00:00, 33940.86it/s]
```

In [20]:

```
preprocessed_titles[20000]
```

Out[20]:

```
'need move input'
```

1.5 Preparing data for models

In [21]:

```
project_data.columns
```

Out[21]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

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

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

Adding the numerical data column to project_data

In [22]:

```
resource_data.head(5)
```

Out[22]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [23]:

```
resource_data.dtypes
```

```
resource_data.dtypes
```

Out[23]:

```
id              object
description     object
quantity         int64
price           object
dtype: object
```

When we dont do the following step what happens is there are a few rows in price and quantity columns which are strings.These cannot be converted to float directly hence the following step.

In [24]:

```
resource_data['price']= pd.to_numeric(resource_data['price'], errors='coerce')

resource_data['quantity'] = pd.to_numeric(resource_data['quantity'], errors='coerce')
```

In [25]:

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

In [26]:

```
project_data.shape
```

Out[26]:

```
(109248, 20)
```

In [27]:

```
project_data.columns
```

Out[27]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'quantity',
       'price'],
      dtype='object')
```

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

In [28]:

```
from sklearn.model_selection import train_test_split
```

In [29]:

```
project_data=project_data[project_data["teacher_prefix"].isna()==False]
project_data_without_label=project_data.loc[:, project_data.columns != 'project_is_approved']
project_is_approved_label=project_data['project_is_approved']
project_is_approved_label=project_is_approved_label.values.reshape(-1,1)
project_data_without_label.head(2)
```

Out[29]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [30]:

```
#https://stackoverflow.com/a/42932524/10939783
# train test split
x_train, x_test, y_train, y_test =
train_test_split(project_data_without_label,project_is_approved_label, test_size=0.20, stratify=pr
oject_is_approved_label)
x_train, x_cv, y_train, y_cv = train_test_split(x_train, y_train, test_size=0.25, stratify=y_train)
print(x_train.shape)
print(y_train.shape)
print(x_cv.shape)
print(y_cv.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(65547, 19)
(65547, 1)
(21849, 19)
(21849, 1)
(21849, 19)
(21849, 1)
```

In [33]:

```
print("Total number of points: ",len(y_train))
print("Number of 1's: ",np.count_nonzero(y_train))
print("Number of 0's: ",len(y_train)-np.count_nonzero(y_train))
```

```
Total number of points:  65547
Number of 1's:  55622
Number of 0's:  9925
```

2.2 Make Data Model Ready: encoding numerical, categorical features

Function to convert to one hot encoding

In [34]:

```
from sklearn.feature_extraction.text import CountVectorizer
def convert_one_hot(dftofit):
    vectorizer = CountVectorizer(lowercase=False, binary=True)
    vectorizer.fit(dftofit.values)
    print(vectorizer.get_feature_names())
    return vectorizer
```

Converting the categories to one hot encoding:

In [35]:

```
#fitting the train data and returning the vectorizer
categories_one_hot_vectorizer=convert_one_hot(x_train['clean_categories'])
```

```
categories_one_hot_vectorizer=convert_one_hot(x_train['clean_categories'])

feature_list=categories_one_hot_vectorizer.get_feature_names()

categories_one_hot_train=categories_one_hot_vectorizer.transform(x_train['clean_categories'].values
)
print(categories_one_hot_train.shape)
categories_one_hot_cv=categories_one_hot_vectorizer.transform(x_cv['clean_categories'].values)
print(categories_one_hot_cv.shape)
categories_one_hot_test=categories_one_hot_vectorizer.transform(x_test['clean_categories'].values)
print(categories_one_hot_test.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
(65547, 9)
(21849, 9)
(21849, 9)
```

Converting the subcategories to one hot encoding:

In [36]:

```
#fitting the train data and returning the vectorizer
subcategories_one_hot_vectorizer=convert_one_hot(x_train['clean_subcategories'])

feature_list.extend(subcategories_one_hot_vectorizer.get_feature_names())

subcategories_one_hot_train=subcategories_one_hot_vectorizer.transform(x_train['clean_subcategories
'].values)
print(subcategories_one_hot_train.shape)
subcategories_one_hot_cv=subcategories_one_hot_vectorizer.transform(x_cv['clean_subcategories'].va
lues)
print(subcategories_one_hot_cv.shape)
subcategories_one_hot_test=subcategories_one_hot_vectorizer.transform(x_test['clean_subcategories'
].values)
print(subcategories_one_hot_test.shape)
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
(65547, 30)
(21849, 30)
(21849, 30)
```

Converting the school_state to one hot encoding:

In [37]:

```
#fitting the train data and returning the vectorizer
school_state_one_hot_vectorizer=convert_one_hot(x_train['school_state'])

feature_list.extend(school_state_one_hot_vectorizer.get_feature_names())

school_state_one_hot_train=school_state_one_hot_vectorizer.transform(x_train['school_state'].values
)
print(school_state_one_hot_train.shape)
school_state_one_hot_cv=school_state_one_hot_vectorizer.transform(x_cv['school_state'].values)
print(school_state_one_hot_cv.shape)
school_state_one_hot_test=school_state_one_hot_vectorizer.transform(x_test['school_state'].values)
print(school_state_one_hot_test.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', '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']
(65547, 51)
(21849, 51)
(21849, 51)
```

Converting the project_grade_category to one hot encoding:

In [38]:

```
x_train['project_grade_category'] = x_train['project_grade_category'].str.replace(" ","_")
x_train['project_grade_category'] = x_train['project_grade_category'].str.replace("-","_")
x_test['project_grade_category'] = x_test['project_grade_category'].str.replace(" ","_")
x_test['project_grade_category'] = x_test['project_grade_category'].str.replace("-","_")

#fitting the train data and returning the vectorizer
project_grade_category_one_hot_vectorizer=convert_one_hot(x_train['project_grade_category'])

feature_list.extend(project_grade_category_one_hot_vectorizer.get_feature_names())

project_grade_category_one_hot_train=project_grade_category_one_hot_vectorizer.transform(x_train['
project_grade_category'].values)
print(project_grade_category_one_hot_train.shape)
project_grade_category_one_hot_cv=project_grade_category_one_hot_vectorizer.transform(x_cv['projec
t_grade_category'].values)
print(project_grade_category_one_hot_cv.shape)
project_grade_category_one_hot_test=project_grade_category_one_hot_vectorizer.transform(x_test['pr
oject_grade_category'].values)
print(project_grade_category_one_hot_test.shape)
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
(65547, 4)
(21849, 4)
(21849, 4)
```

Converting the teacher_prefix to one hot encoding:

In [39]:

```
#fitting the train data and returning the vectorizer
teacher_prefix_one_hot_vectorizer=convert_one_hot(x_train['teacher_prefix'])

feature_list.extend(teacher_prefix_one_hot_vectorizer.get_feature_names())

teacher_prefix_one_hot_train=teacher_prefix_one_hot_vectorizer.transform(x_train['teacher_prefix']
.values)
print(teacher_prefix_one_hot_train.shape)
teacher_prefix_one_hot_cv=teacher_prefix_one_hot_vectorizer.transform(x_cv['teacher_prefix'].value
s)
print(teacher_prefix_one_hot_cv.shape)
teacher_prefix_one_hot_test=teacher_prefix_one_hot_vectorizer.transform(x_test['teacher_prefix'].v
alues)
print(teacher_prefix_one_hot_test.shape)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
(65547, 5)
(21849, 5)
(21849, 5)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

Preprocessing the titles and essays:

Preprocessing the essays of train data:

In [40]:

```
from tqdm import tqdm
preprocessed_essays_train_data = []
# tqdm is for printing the status bar
for sentance in tqdm(x_train['essay'].values):
```

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

```
100%|██████████| 65547/65547 [00:44<00:00, 1486.59it/s]
```

Preprocessing the essays of cv data:

In [41]:

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

```
100%|██████████| 21849/21849 [00:14<00:00, 1498.57it/s]
```

Preprocessing the essays of test data:

In [42]:

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

```
100%|██████████| 21849/21849 [00:14<00:00, 1501.45it/s]
```

Preprocessing the titles of train data:

In [43]:

```
from tqdm import tqdm
preprocessed_titles_train_data = []
# tqdm is for printing the status bar
for sentance in tqdm(x_train['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)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles_train_data.append(sent.lower().strip())
```

```
100%|██████████| 65547/65547 [00:01<00:00, 33603.34it/s]
```

Preprocessing the titles of cv data:

In [44]:

```
from tqdm import tqdm
preprocessed_titles_cv_data = []
# tqdm is for printing the status bar
for sentance in tqdm(x_cv['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)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles_cv_data.append(sent.lower().strip())
```

```
100%|██████████| 21849/21849 [00:00<00:00, 33043.01it/s]
```

Preprocessing the titles of test data:

In [45]:

```
from tqdm import tqdm
preprocessed_titles_test_data = []
# tqdm is for printing the status bar
for sentance in tqdm(x_test['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)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles_test_data.append(sent.lower().strip())
```

```
100%|██████████| 21849/21849 [00:00<00:00, 32890.24it/s]
```

Bag of words on essay and titles:

In [46]:

```
# function to fit using train data
def do_bow(dfbow):
    vectorizer=CountVectorizer(min_df=10,max_features=5000,ngram_range=(1,2))
    vectorizer.fit(dfbow)
    return vectorizer
# function to fit using train data
def do_bow_titles(dfbow):
    vectorizer=CountVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
    vectorizer.fit(dfbow)
    return vectorizer
essay_bow_vectorizer=do_bow(preprocessed_essays_train_data)
print(type(essay_bow_vectorizer))

feature_list.extend(essay_bow_vectorizer.get_feature_names())

train_essay_bow=essay_bow_vectorizer.transform(preprocessed_essays_train_data)
print(train_essay_bow.shape)

cv_essay_bow=essay_bow_vectorizer.transform(preprocessed_essays_cv_data)
print(cv_essay_bow.shape)

test_essay_bow=essay_bow_vectorizer.transform(preprocessed_essays_test_data)
print(test_essay_bow.shape)

titles_bow_vectorizer=do_bow_titles(preprocessed_titles_train_data)
print(type(titles_bow_vectorizer))

feature_list.extend(titles_bow_vectorizer.get_feature_names())
```

```
train_titles_bow=titles_bow_vectorizer.transform(preprocessed_titles_train_data)
print(train_titles_bow.shape)

cv_titles_bow=titles_bow_vectorizer.transform(preprocessed_titles_cv_data)
print(cv_titles_bow.shape)

test_titles_bow=titles_bow_vectorizer.transform(preprocessed_titles_test_data)
print(test_titles_bow.shape)

print(len(feature_list))
```

```
<class 'sklearn.feature_extraction.text.CountVectorizer'>
(65547, 5000)
(21849, 5000)
(21849, 5000)
<class 'sklearn.feature_extraction.text.CountVectorizer'>
(65547, 4141)
(21849, 4141)
(21849, 4141)
9240
```

TFIDF on essay and titles

In [47]:

```
# function to fit using train data
def do_tfidf(dfbow):
    vectorizer=TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
    vectorizer.fit(dfbow)
    return vectorizer
def do_tfidf_titles(dfbow):
    vectorizer=TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
    vectorizer.fit(dfbow)
    return vectorizer
essay_tfidf_vectorizer=do_tfidf(preprocessed_essays_train_data)
print(type(essay_bow_vectorizer))
train_essay_tfidf=essay_tfidf_vectorizer.transform(preprocessed_essays_train_data)
print(train_essay_tfidf.shape)
cv_essay_tfidf=essay_tfidf_vectorizer.transform(preprocessed_essays_cv_data)
print(cv_essay_tfidf.shape)
test_essay_tfidf=essay_tfidf_vectorizer.transform(preprocessed_essays_test_data)
print(test_essay_tfidf.shape)

titles_tfidf_vectorizer=do_tfidf_titles(preprocessed_titles_train_data)
print(type(titles_tfidf_vectorizer))
train_titles_tfidf=titles_tfidf_vectorizer.transform(preprocessed_titles_train_data)
print(train_titles_tfidf.shape)
cv_titles_tfidf=titles_tfidf_vectorizer.transform(preprocessed_titles_cv_data)
print(cv_titles_tfidf.shape)
test_titles_tfidf=titles_tfidf_vectorizer.transform(preprocessed_titles_test_data)
print(test_titles_tfidf.shape)
```

```
<class 'sklearn.feature_extraction.text.CountVectorizer'>
(65547, 5000)
(21849, 5000)
(21849, 5000)
<class 'sklearn.feature_extraction.text.TfidfVectorizer'>
(65547, 4141)
(21849, 4141)
(21849, 4141)
```

Vectorising the numerical features

In [48]:

```
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73.   5.5 ].
```

```
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
train_price_standardized = price_scalar.transform(x_train['price'].values.reshape(-1, 1))
cv_price_standardized = price_scalar.transform(x_cv['price'].values.reshape(-1, 1))
test_price_standardized = price_scalar.transform(x_test['price'].values.reshape(-1, 1))

import warnings
warnings.filterwarnings("ignore")
teacher_number_of_previously_posted_projects_scalar=StandardScaler()
teacher_number_of_previously_posted_projects_scalar.fit(x_train.teacher_number_of_previously_posted
projects.values.reshape(-1,1))

print("Mean : ",teacher_number_of_previously_posted_projects_scalar.mean_[0], "Standard deviation
: ",np.sqrt(teacher_number_of_previously_posted_projects_scalar.var_[0]))

train_teacher_number_of_previously_posted_projects_standardized=teacher_number_of_previously_posted
projects_scalar.transform(x_train.teacher_number_of_previously_posted_projects.values.reshape(-1,1
))
cv_teacher_number_of_previously_posted_projects_standardized=teacher_number_of_previously_posted_pr
jects_scalar.transform(x_cv.teacher_number_of_previously_posted_projects.values.reshape(-1,1))
test_teacher_number_of_previously_posted_projects_standardized=teacher_number_of_previously_posted_
rojects_scalar.transform(x_test.teacher_number_of_previously_posted_projects.values.reshape(-1,1))
```

```
Mean :  879.7708480937342 Standard deviation : 1189.081052080077
Mean :  11.151768959677788 Standard deviation :  27.69288334196712
```

In [62]:

```
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
teacher_number_of_previously_posted_projects_minmaxscalar= MinMaxScaler()
teacher_number_of_previously_posted_projects_minmaxscalar.fit(x_train.teacher_number_of_previously_
osted_projects.values.reshape(-1,1))
train_teacher_number_of_previously_posted_projects_normalised=teacher_number_of_previously_posted_p
ojects_minmaxscalar.transform(x_train.teacher_number_of_previously_posted_projects.values.reshape(
-1,1))
cv_teacher_number_of_previously_posted_projects_normalised=teacher_number_of_previously_posted_proj
cts_minmaxscalar.transform(x_cv.teacher_number_of_previously_posted_projects.values.reshape(-1,1))
test_teacher_number_of_previously_posted_projects_normalised=teacher_number_of_previously_posted_pr
jects_minmaxscalar.transform(x_test.teacher_number_of_previously_posted_projects.values.reshape(-1
,1))
```

In [63]:

```
price_minmaxscalar = MinMaxScaler()
price_minmaxscalar.fit(x_train['price'].values.reshape(-1,1))
train_price_normalised = price_minmaxscalar.transform(x_train['price'].values.reshape(-1, 1))
cv_price_normalised = price_minmaxscalar.transform(x_cv['price'].values.reshape(-1, 1))
test_price_normalised = price_minmaxscalar.transform(x_test['price'].values.reshape(-1, 1))
```

Average word to vector on essays and title

In [49]:

```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
```

```
model = loadGloveModel('glove.42B.300d.txt')
words = []
for i in preprocessed_essays_train_data:
    words.extend(i.split(' '))

for i in preprocessed_titles_train_data:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

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

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


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

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

```
402it [00:00, 3123.61it/s]
```

```
Loading Glove Model
```

```
298709it [00:37, 8015.73it/s]
```

```
Done. 298709  words loaded!
all the words in the coupus 9292902
the unique words in the coupus 48170
The number of words that are present in both glove vectors and our coupus 38722 ( 80.386 %)
word 2 vec length 38722
```

In [59]:

```
# 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())
```

Avg w2v for essays used for training

In [51]:

```
train_essay_avgw2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_train_data): # 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
    train_essay_avgw2v.append(vector)

print(len(train_essay_avgw2v))
print(len(train_essay_avgw2v[0]))
```

```
100%|██████████| 65547/65547 [00:23<00:00, 2841.84it/s]
```

```
65547
300
```

Avg w2v for essays used for cv

In [52]:

```
cv_essay_avgw2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_cv_data): # 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
    cv_essay_avgw2v.append(vector)

print(len(cv_essay_avgw2v))
print(len(cv_essay_avgw2v[0]))
```

```
100%|██████████| 21849/21849 [00:07<00:00, 2807.52it/s]
```

```
21849
300
```

Avg w2v for essays used for testing

In [53]:

```
test_essay_avgw2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_test_data): # 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
    test_essay_avgw2v.append(vector)

print(len(test_essay_avgw2v))
print(len(test_essay_avgw2v[0]))
```

```
100%|██████████| 21849/21849 [00:07<00:00, 2933.39it/s]
```

```
21849
300
```

In [54]:

```
titlewords = []


for i in preprocessed_titles:
    titlewords.extend(i.split(' '))
print("all the words in the coupus", len(titlewords))
titlewords = set(titlewords)
print("the unique words in the coupus", len(titlewords))

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

titlewords_courpus = {}
words_glove = set(model.keys())
```

```
for i in titlewords:
    if i in words_glove:
        titlewords_courpus[i] = model[i]
print("word 2 vec length", len(titlewords_courpus))


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

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(titlewords_courpus, f)
# 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)
    titleglove_words =  set(model.keys())
```

```
all the words in the coupus 403696
the unique words in the coupus 16773
The number of words that are present in both glove vectors and our coupus 14060 ( 83.825 %)
word 2 vec length 14060
```

Avg w2v for titles used for training

In [55]:

```
train_titles_avgw2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_train_data): # 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 titleglove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_titles_avgw2v.append(vector)

print(len(train_titles_avgw2v))
print(len(train_titles_avgw2v[0]))
```

```
100%|██████████| 65547/65547 [00:01<00:00, 59043.41it/s]
```

```
65547
300
```

Avg w2v for titles used for cv

In [56]:

```
cv_titles_avgw2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_cv_data): # 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 titleglove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    cv_titles_avgw2v.append(vector)

print(len(cv_titles_avgw2v))
print(len(cv_titles_avgw2v[0]))
```

```
100%|██████████| 21849/21849 [00:00<00:00, 59572.12it/s]
```

```
21849
```

```
300
```

Avg w2v for titles used for testing

In [57]:

```
test_titles_avgw2v = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_test_data): # 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 titleglove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_titles_avgw2v.append(vector)

print(len(test_titles_avgw2v))
print(len(test_titles_avgw2v[0]))
```

```
100%|██████████| 21849/21849 [00:00<00:00, 59899.40it/s]

21849
300
```

Tfidf Average word to vector on essays and title

In [60]:

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

def do_tfidf_w2v(preprocessed_essays):
    tfidf_w2v_vectors=[]
    for sentence in tqdm(preprocessed_essays): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        tf_idf_weight =0; # num of words with a valid vector in the sentence/review
        for word in sentence.split(): # for each word in a review/sentence
            if (word in glove_words) and (word in tfidf_words):
                vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((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
        tfidf_w2v_vectors.append(vector)
    return tfidf_w2v_vectors

train_essay_tfidf_w2v_vectors=do_tfidf_w2v(preprocessed_essays_train_data)
cv_essay_tfidf_w2v_vectors=do_tfidf_w2v(preprocessed_essays_cv_data)
test_essay_tfidf_w2v_vectors=do_tfidf_w2v(preprocessed_essays_test_data)
print(len(train_essay_tfidf_w2v_vectors))
print(len(train_essay_tfidf_w2v_vectors[0]))
print(len(cv_essay_tfidf_w2v_vectors))
print(len(cv_essay_tfidf_w2v_vectors[0]))
print(len(test_essay_tfidf_w2v_vectors))
print(len(test_essay_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 65547/65547 [02:23<00:00, 456.88it/s]
100%|██████████| 21849/21849 [00:46<00:00, 471.74it/s]
100%|██████████| 21849/21849 [00:45<00:00, 475.68it/s]
```

```
65547
300
21849
300
21849
300
```

In [61]:

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

def do_tfidf_w2v(preprocessed_essays):
    tfidf_w2v_vectors=[]
    for sentence in tqdm(preprocessed_essays): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        tf_idf_weight =0; # num of words with a valid vector in the sentence/review
        for word in sentence.split(): # for each word in a review/sentence
            if (word in titleglove_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
        tfidf_w2v_vectors.append(vector)
    return tfidf_w2v_vectors

train_titles_tfidf_w2v_vectors=do_tfidf_w2v(preprocessed_titles_train_data)
cv_titles_tfidf_w2v_vectors=do_tfidf_w2v(preprocessed_titles_cv_data)
test_titles_tfidf_w2v_vectors=do_tfidf_w2v(preprocessed_titles_test_data)
print(len(train_titles_tfidf_w2v_vectors))
print(len(train_titles_tfidf_w2v_vectors[0]))
print(len(cv_titles_tfidf_w2v_vectors))
print(len(cv_titles_tfidf_w2v_vectors[0]))
print(len(test_titles_tfidf_w2v_vectors))
print(len(test_titles_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 65547/65547 [00:02<00:00, 29347.47it/s]
100%|██████████| 21849/21849 [00:00<00:00, 26383.46it/s]
100%|██████████| 21849/21849 [00:00<00:00, 30376.81it/s]
```

```
65547
300
21849
300
21849
300
```

Stacking together all the variables of train data

In [64]:

```
from scipy.sparse import hstack
train_set_1=hstack((categories_one_hot_train,subcategories_one_hot_train,school_state_one_hot_trair
,project_grade_category_one_hot_train,teacher_prefix_one_hot_train,train_essay_bow,train_titles_bow
,train_price_standardized,train_teacher_number_of_previously_posted_projects_standardized)).tocsr(
)
train_set_2=hstack((categories_one_hot_train,subcategories_one_hot_train,school_state_one_hot_trair
,project_grade_category_one_hot_train,teacher_prefix_one_hot_train,train_essay_tfidf,train_titles_t
fidf,train_price_standardized,train_teacher_number_of_previously_posted_projects_standardized)).to
csr()
train_set_3=hstack((categories_one_hot_train,subcategories_one_hot_train,school_state_one_hot_trair
,project_grade_category_one_hot_train,teacher_prefix_one_hot_train,train_essay_avgw2v,train_titles_
avgw2v,train_price_standardized,train_teacher_number_of_previously_posted_projects_standardized)).
```

```
tocsr()
train_set_4=hstack((categories_one_hot_train,subcategories_one_hot_train,school_state_one_hot_trair
,project_grade_category_one_hot_train,teacher_prefix_one_hot_train,train_essay_tfidf_w2v_vectors,t
rain_titles_tfidf_w2v_vectors,train_price_standardized,train_teacher_number_of_previously_posted_pr
jects_standardized)).tocsr()
print(train_set_1.shape)
print(train_set_2.shape)
print(train_set_3.shape)
print(train_set_4.shape)
cv_set_1=hstack((categories_one_hot_cv,subcategories_one_hot_cv,school_state_one_hot_cv,project_gra
de_category_one_hot_cv,teacher_prefix_one_hot_cv,cv_essay_bow,cv_titles_bow,cv_price_standardized,
cv_teacher_number_of_previously_posted_projects_standardized)).tocsr()
cv_set_2=hstack((categories_one_hot_cv,subcategories_one_hot_cv,school_state_one_hot_cv,project_gra
de_category_one_hot_cv,teacher_prefix_one_hot_cv,cv_essay_tfidf,cv_titles_tfidf,cv_price_standardiz
ed,cv_teacher_number_of_previously_posted_projects_standardized)).tocsr()
cv_set_3=hstack((categories_one_hot_cv,subcategories_one_hot_cv,school_state_one_hot_cv,project_gra
de_category_one_hot_cv,teacher_prefix_one_hot_cv,cv_essay_avgw2v,cv_titles_avgw2v,cv_price_standarc
ized,cv_teacher_number_of_previously_posted_projects_standardized)).tocsr()
cv_set_4=hstack((categories_one_hot_cv,subcategories_one_hot_cv,school_state_one_hot_cv,project_gra
de_category_one_hot_cv,teacher_prefix_one_hot_cv,cv_essay_tfidf_w2v_vectors,cv_titles_tfidf_w2v_vec
tors,cv_price_standardized,cv_teacher_number_of_previously_posted_projects_standardized)).tocsr()
print(cv_set_1.shape)
print(cv_set_2.shape)
print(cv_set_3.shape)
print(cv_set_4.shape)
test_set_1=hstack((categories_one_hot_test,subcategories_one_hot_test,school_state_one_hot_test,pr
oject_grade_category_one_hot_test,teacher_prefix_one_hot_test,test_essay_bow,test_titles_bow,test_
price_standardized,test_teacher_number_of_previously_posted_projects_standardized)).tocsr()
test_set_2=hstack((categories_one_hot_test,subcategories_one_hot_test,school_state_one_hot_test,pr
oject_grade_category_one_hot_test,teacher_prefix_one_hot_test,test_essay_tfidf,test_titles_tfidf,t
est_price_standardized,test_teacher_number_of_previously_posted_projects_standardized)).tocsr()
test_set_3=hstack((categories_one_hot_test,subcategories_one_hot_test,school_state_one_hot_test,pr
oject_grade_category_one_hot_test,teacher_prefix_one_hot_test,test_essay_avgw2v,test_titles_avgw2v
,test_price_standardized,test_teacher_number_of_previously_posted_projects_standardized)).tocsr()
test_set_4=hstack((categories_one_hot_test,subcategories_one_hot_test,school_state_one_hot_test,pr
oject_grade_category_one_hot_test,teacher_prefix_one_hot_test,test_essay_tfidf_w2v_vectors,test_ti
tles_tfidf_w2v_vectors,test_price_standardized,test_teacher_number_of_previously_posted_projects_st
dardized)).tocsr()
print(test_set_1.shape)
print(test_set_2.shape)
print(test_set_3.shape)
print(test_set_4.shape)
```

```
(65547, 9207)
(65547, 9207)
(65547, 701)
(65547, 701)
(21849, 9207)
(21849, 9207)
(21849, 701)
(21849, 701)
(21849, 9207)
(21849, 9207)
(21849, 701)
(21849, 701)
```

In [71]:

```
#https://stackoverflow.com/questions/8955448/save-load-scipy-sparse-csr-matrix-in-portable-data-fo
rmat
from scipy import sparse
sparse.save_npz("train_set_1.npz", train_set_1)
sparse.save_npz("train_set_2.npz", train_set_2)
sparse.save_npz("train_set_3.npz", train_set_3)
sparse.save_npz("train_set_4.npz", train_set_4)
sparse.save_npz("cv_set_1.npz", cv_set_1)
sparse.save_npz("cv_set_2.npz", cv_set_2)
sparse.save_npz("cv_set_3.npz", cv_set_3)
sparse.save_npz("cv_set_4.npz", cv_set_4)
sparse.save_npz("test_set_1.npz", test_set_1)
sparse.save_npz("test_set_2.npz", test_set_2)
sparse.save_npz("test_set_3.npz", test_set_3)
sparse.save_npz("test_set_4.npz", test_set_4)
```

In [67]:

```
train_set_1_saved = sparse.load_npz("train_set_1.npz")
```

In [70]:

```
print(train_set_1_saved.shape)
```

```
(65547, 9207)
```

2.4 Appling KNN on different kind of featurizations

2.4.1 Applying KNN brute force on BOW, SET 1

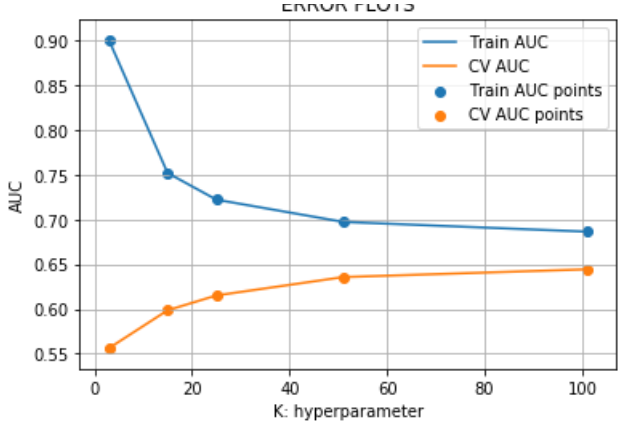
In [74]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000

    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [126]:

```
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]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(train_set_1,y_train)

    y_train_pred = batch_predict(neigh,train_set_1)
    y_cv_pred = batch_predict(neigh,cv_set_1)

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

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

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

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

```
100%|██████████| 5/5 [53:36<00:00, 642.33s/it]
```



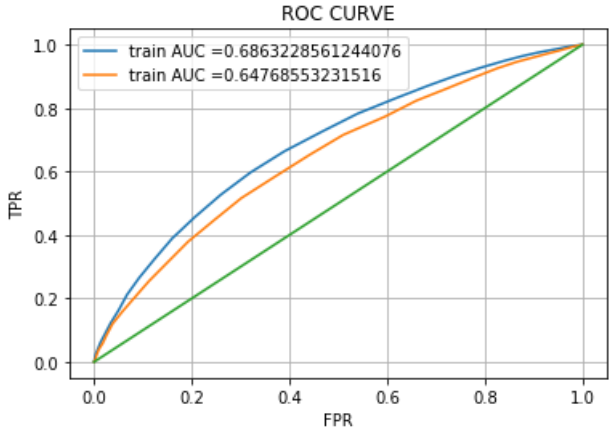
We can see that the best possible K from the above plot is 101.

In [127]:

```
y_test_pred = batch_predict(neigh,test_set_1)
```

In [128]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,y_test_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.plot(train_fpr,train_fpr)
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



We can see that with K=101 the classification capability of this classifier is pretty good.

Finding the best threshold

In [73]:

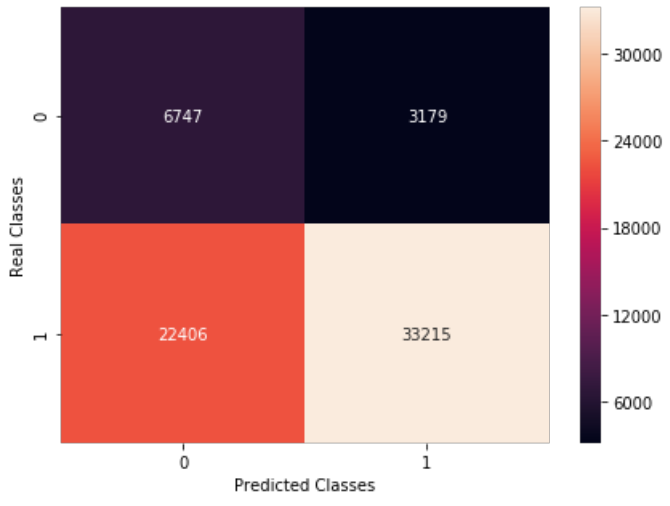
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
```

```
    # (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
```

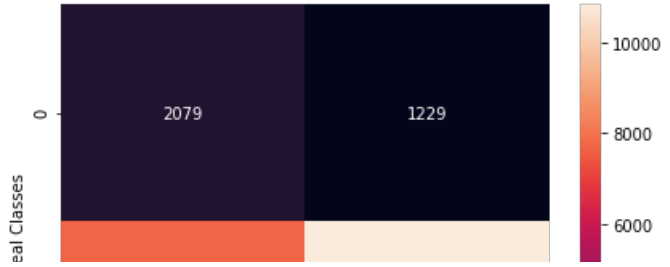
In [130]:

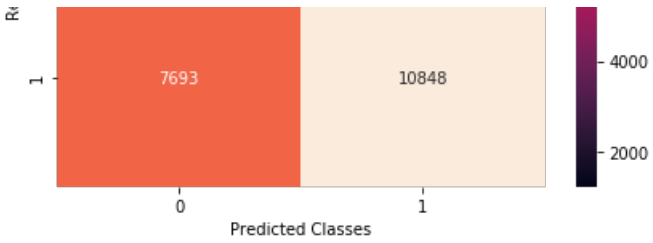
```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred,
best_t)),range(2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
print("Test confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)),range(
2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.405912011954572 for threshold 0.822
Train confusion matrix
```




```
Test confusion matrix
```





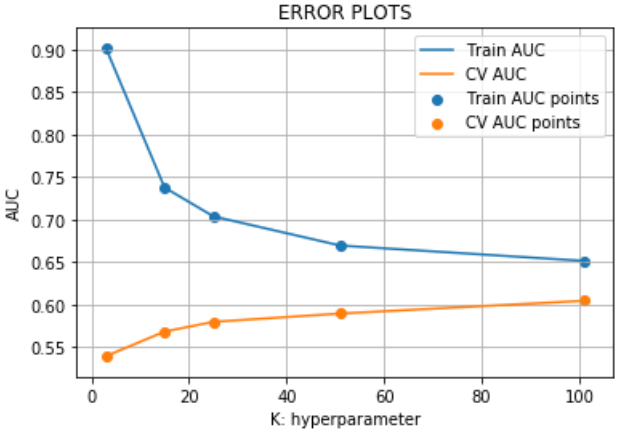
The number of true positives we are getting is high,but at the same time we are also getting a large number of false negatives.

2.4.2 Applying KNN brute force on TFIDF, SET 2

In [132]:

```
train_auc = []
cv_auc = []
K = [3, 15, 25, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(train_set_2,y_train)

    y_train_pred = batch_predict(neigh,train_set_2)
    y_cv_pred = batch_predict(neigh,cv_set_2)

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

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

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

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

```
  0%|          | 0/5 [00:00<?, ?it/s]
 20%|██        | 1/5 [10:14<40:57, 614.27s/it]
 40%|████      | 2/5 [21:08<31:18, 626.26s/it]
 60%|██████    | 3/5 [32:00<21:07, 633.84s/it]
 80%|████████  | 4/5 [42:50<10:38, 638.85s/it]
100%|██████████| 5/5 [53:42<00:00, 642.66s/it]
```



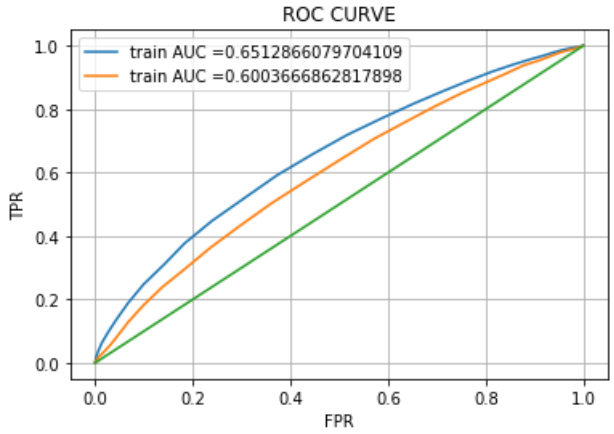
Here also the best possible K is 101.

In [133]:

```
y_test_pred = batch_predict(neigh,test_set_2)
```

In [134]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,y_test_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.plot(train_fpr,train_fpr)
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



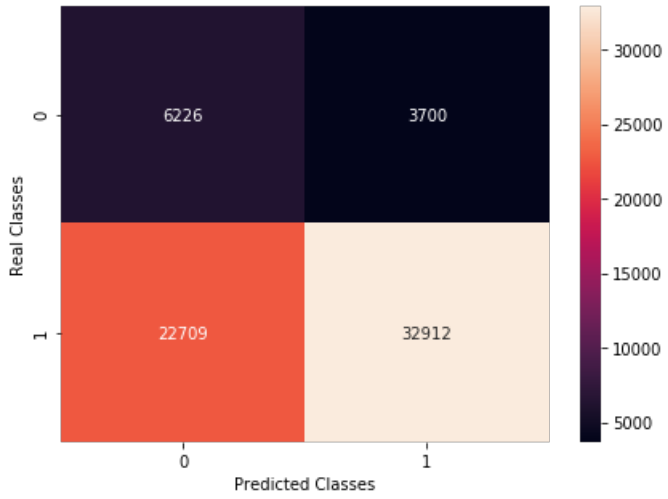
Classification is being done decently.

In [135]:

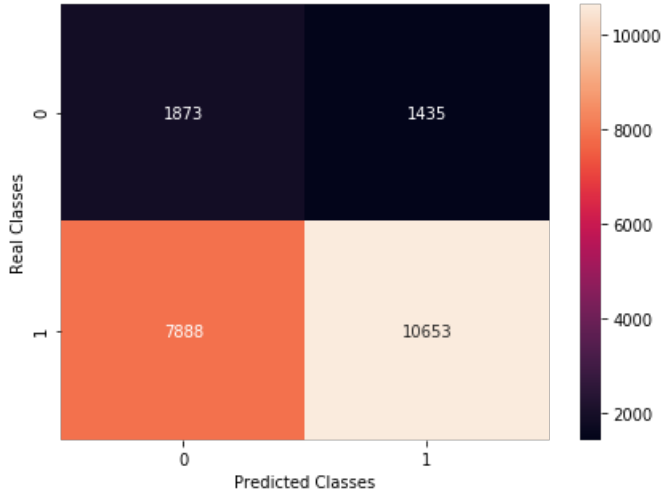
```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred,
best_t)),range(2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
print("Test confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)),range(
2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
```

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

the maximum value of tpr*(1-fpr) 0.37115073688007133 for threshold 0.851
```

Train confusion matrix



Test confusion matrix



The false negative rate looks high in this case as well.

K-Best features on set-2 using chi-square

The price and number of previously posted projects have been normalised in this case because standardisation gives value beteen -1 and 1 which won't work for chi square,so we use normalisation.

In [31]:

```
from scipy import sparse
train_set_2 = sparse.load_npz("train_set_2.npz")
cv_set_2 = sparse.load_npz("cv_set_2.npz")
test_set_2 = sparse.load_npz("test_set_2.npz")
```

In [65]:

```
from scipy.sparse import hstack
train_set_2_normal=hstack((categories_one_hot_train,subcategories_one_hot_train,school_state_one_ho
t_train,project_grade_category_one_hot_train,teacher_prefix_one_hot_train,train_essay_tfidf,train_
titles_tfidf,train_price_normalised,train_teacher_number_of_previously_posted_projects_normalised)
).tocsr()
```

In [68]:

```
cv_set_2_normal=hstack((categories_one_hot_cv,subcategories_one_hot_cv,school_state_one_hot_cv,pro
```

```
ject_grade_category_one_hot_cv,teacher_prefix_one_hot_cv,cv_essay_tfidf,cv_titles_tfidf,cv_price_no
rmalised,cv_teacher_number_of_previously_posted_projects_normalised)).tocsr()
test_set_2_normal=hstack((categories_one_hot_test,subcategories_one_hot_test,school_state_one_hot_t
est,project_grade_category_one_hot_test,teacher_prefix_one_hot_test,test_essay_tfidf,test_titles_tf
idf,test_price_normalised,test_teacher_number_of_previously_posted_projects_normalised)).tocsr()
```

In [66]:

```
from sklearn.feature_selection import SelectKBest, chi2
featuremodelchi2 = SelectKBest(chi2, k=2000).fit(train_set_2_normal,y_train)
```

In [70]:

```
X_new = featuremodelchi2.transform(train_set_2_normal)
X_new.shape
```

Out[70]:

```
(65547, 2000)
```

In [71]:

```
X_cv_2_new = featuremodelchi2.transform(cv_set_2_normal)
X_cv_2_new.shape
```

Out[71]:

```
(21849, 2000)
```

In [76]:

```
X_test_2_new = featuremodelchi2.transform(test_set_2_normal)
X_test_2_new.shape
```

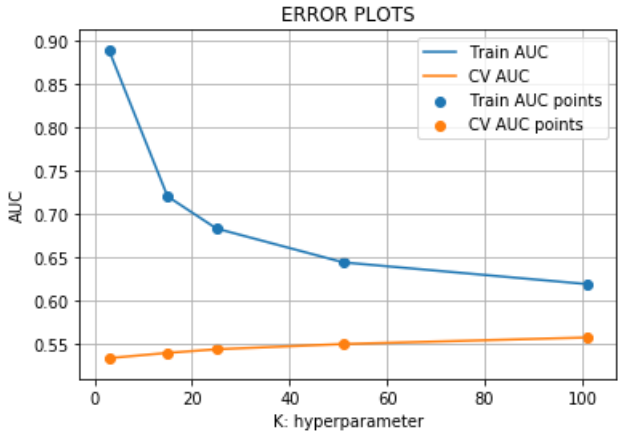
Out[76]:

```
(21849, 2000)
```

In [75]:

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

    y_train_pred = batch_predict(neigh,X_new)
    y_cv_pred = batch_predict(neigh,X_cv_2_new)

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

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

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

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

```
100%|██████████| 5/5 [47:48<00:00, 573.06s/it]
```



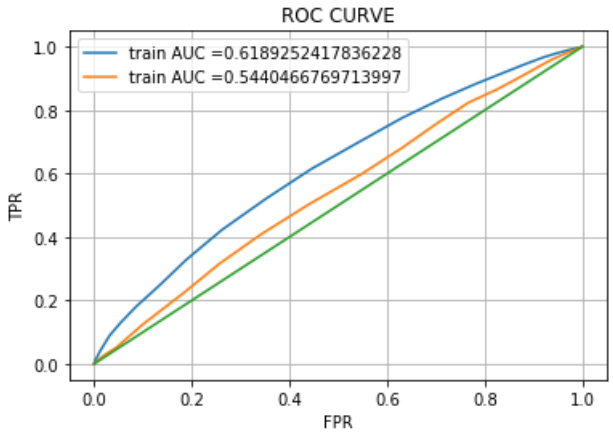
We can again take k to be a value around 101.

In [77]:

```
y_test_pred = batch_predict(neigh,X_test_2_new)
```

In [78]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,y_test_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.plot(train_fpr,train_fpr)
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



Classication seems good.

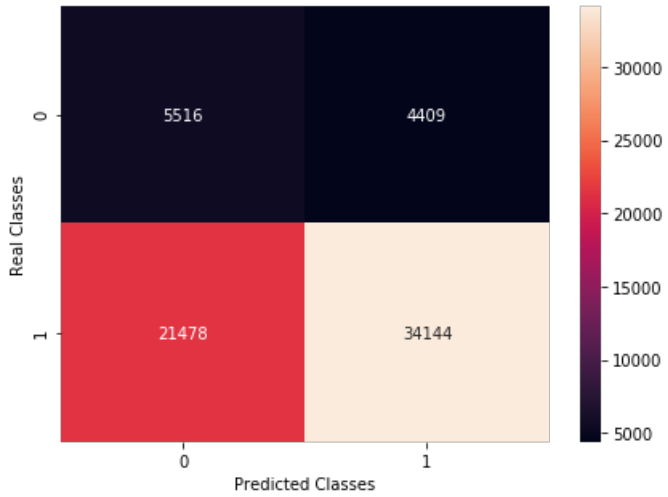
In [79]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
```

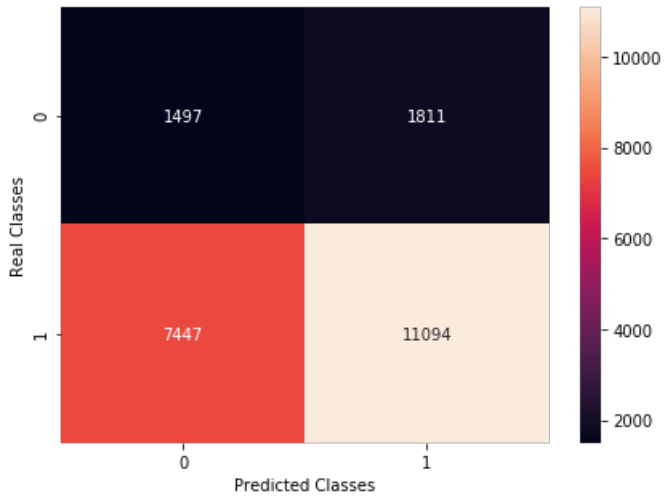
```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred,
best_t)),range(2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
print("Test confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)),range(
2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
```

==========

```
the maximum value of tpr*(1-fpr) 0.34116269707173297 for threshold 0.842
Train confusion matrix
```




```
Test confusion matrix
```



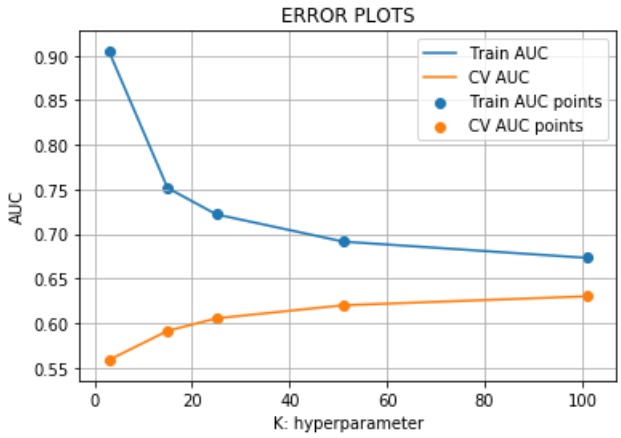
Again false negative rate is high in this case.

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

In [74]:

```
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]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(train_set_3,y_train)

    y_train_pred = batch_predict(neigh,train_set_3)
    y_cv_pred = batch_predict(neigh,cv_set_3)

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

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

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

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

```
100%|██████████| 5/5 [3:28:10<00:00, 2499.12s/it]
```



K is 101 in this case also.

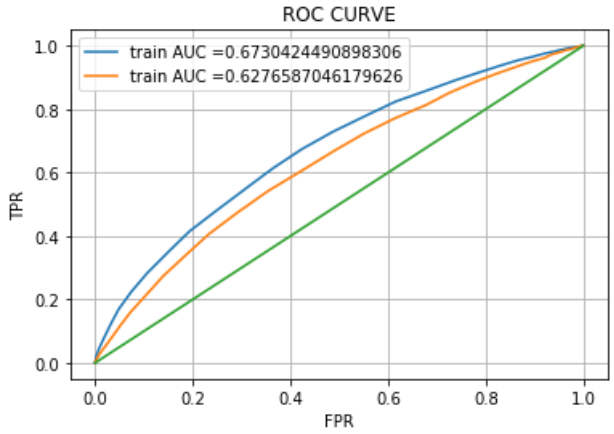
In [75]:

```
y_test_pred = batch_predict(neigh,test_set_3)
```

In [76]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,y_test_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.plot(train_fpr,train_fpr)
```

```
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```

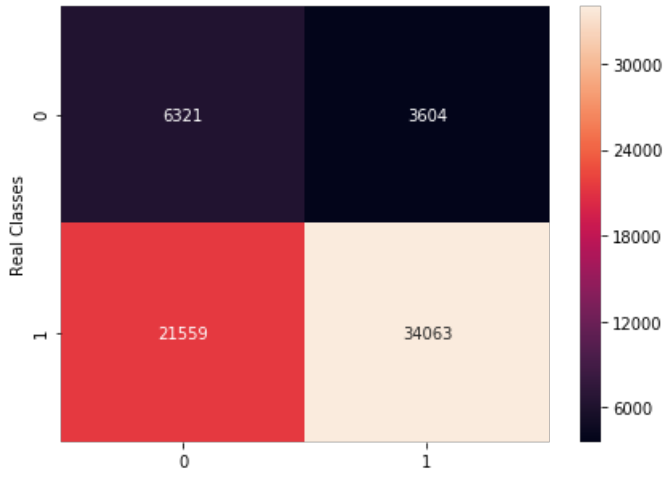


Classification is good in this scenario also.

In [79]:

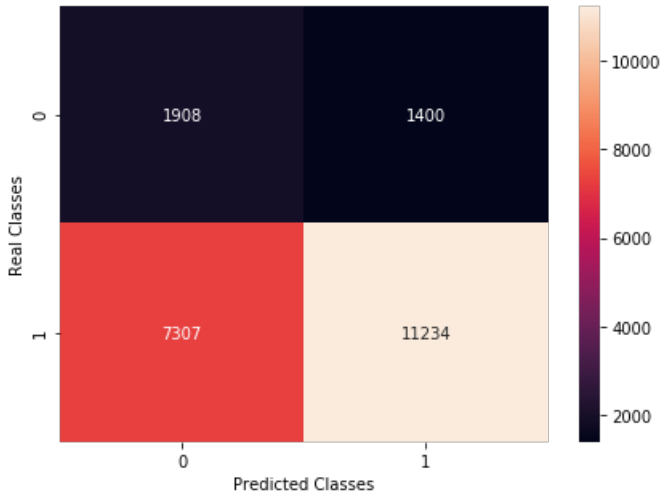
```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred,
best_t)),range(2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
print("Test confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)),range(
2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
```

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

the maximum value of tpr*(1-fpr) 0.3900242125531215 for threshold 0.851
Train confusion matrix
```



Predicted Classes

Test confusion matrix



The false positive rate is again high in this case.

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

In [5]:

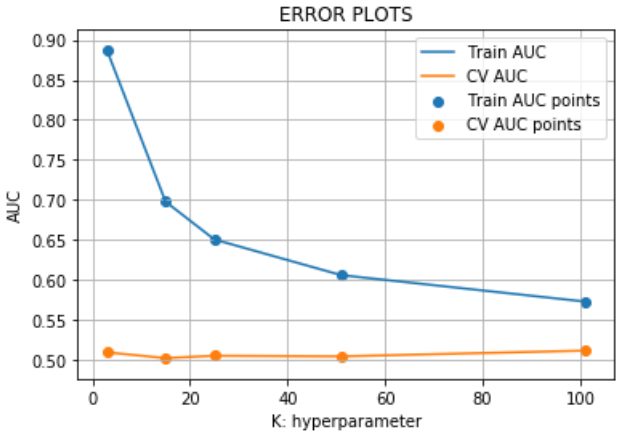
```
from scipy import sparse
train_set_4 = sparse.load_npz("train_set_4.npz")
cv_set_4 = sparse.load_npz("cv_set_4.npz")
test_set_4 = sparse.load_npz("test_set_4.npz")
```

In [43]:

```
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]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(train_set_4,y_train)

    y_train_pred = batch_predict(neigh,train_set_4)
    y_cv_pred = batch_predict(neigh,cv_set_4)

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

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

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

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

```
  0%|          | 0/5 [00:00<?, ?it/s]
 20%|██        | 1/5 [39:43<2:38:52, 2383.06s/it]
```

```
 40%|████        | 2/5 [1:20:01<1:59:40, 2393.66s/it]
 60%|██████      | 3/5 [2:00:14<1:19:59, 2399.53s/it]
 80%|████████    | 4/5 [2:40:32<40:05, 2405.14s/it]
100%|██████████| 5/5 [3:20:53<00:00, 2409.71s/it]
```



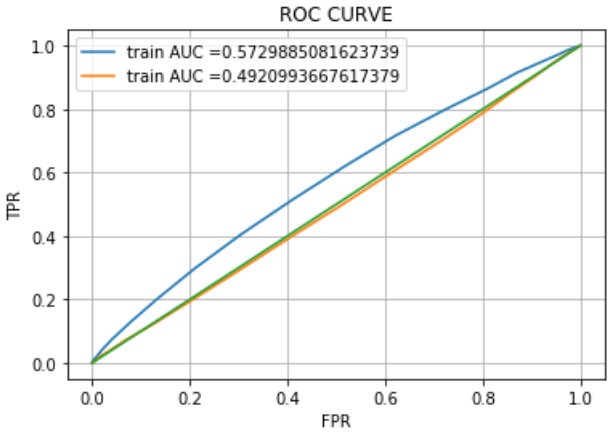
We will choose

In [45]:

```
y_test_pred = batch_predict(neigh,test_set_4)
```

In [46]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train,y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test,y_test_pred)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.plot(train_fpr,train_fpr)
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



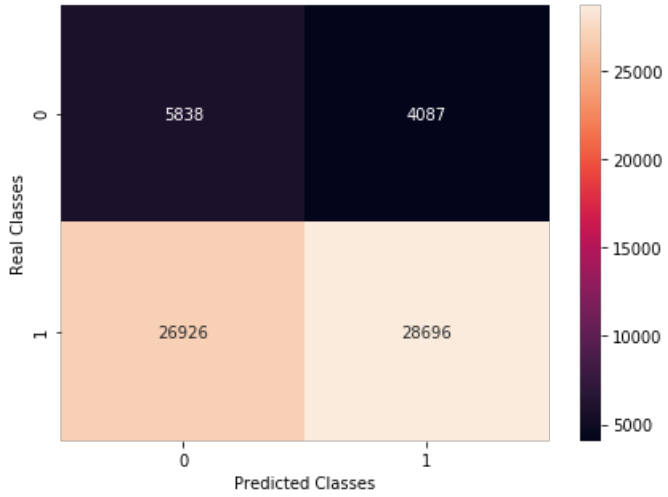
In [47]:

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

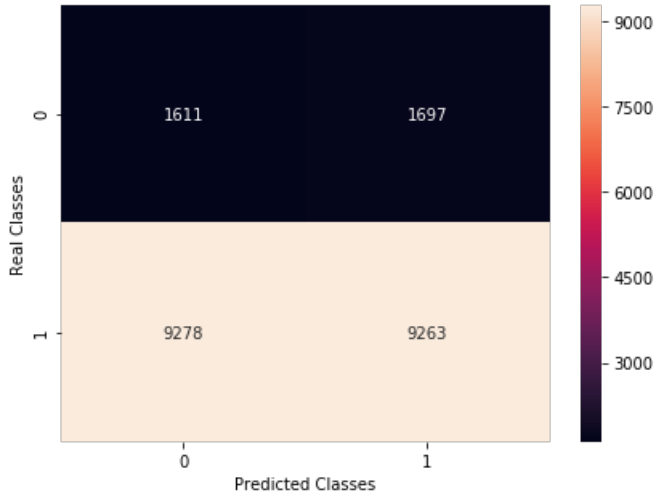
```
print("Train confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred,
best_t)),range(2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
print("Test confusion matrix")
conf_matrix= pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)),range(
2),range(2))
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(1,2,2)
sns.heatmap(conf_matrix,annot=True,ax=ax,fmt='g')
plt.ylabel('Real Classes')
plt.xlabel('Predicted Classes')
plt.show()
```

==========

```
the maximum value of tpr*(1-fpr) 0.3034648106456618 for threshold 0.851
Train confusion matrix
```




```
Test confusion matrix
```



CONCLUSIONS:

In [1]:

```
from prettytable import PrettyTable

x = PrettyTable()
```

```
x.field_names = ["Vectorizer", "Model","Train-AUC", "Test-AUC"]

x.add_row(["BOW", "KNN", 0.686,0.647 ])
x.add_row(["TFIDF", "KNN", 0.651, 0.600])
x.add_row(["TFIDF with K-best features", "KNN", 0.618, 0.544])
x.add_row(["AVG-W2V","KNN", 0.673,0.627 ])
x.add_row(["TFIDF-W2V", "KNN", 0.572, 0.492])

print(x)
```

```
+----------------------------+-------+-----------+----------+
|         Vectorizer         | Model | Train-AUC | Test-AUC |
+----------------------------+-------+-----------+----------+
|            BOW             |  KNN  |   0.686   |  0.647   |
|           TFIDF            |  KNN  |   0.651   |   0.6    |
| TFIDF with K-best features |  KNN  |   0.618   |  0.544   |
|          AVG-W2V           |  KNN  |   0.673   |  0.627   |
|         TFIDF-W2V          |  KNN  |   0.572   |  0.492   |
+----------------------------+-------+-----------+----------+
```