Applying k NN on Donors Choose 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` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **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's title. One of the following enumerated values: • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_prefix	
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same teacher. **Example:** 2

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_4:__ "Close by sharing why your project will make a difference"

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

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

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

In [2]:

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

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

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

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

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

from tqdm import tqdm
import os

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

Reading Data

In [3]:

```
project_data = pd.read_csv('train_data.csv', nrows=50000)
resource_data = pd.read_csv('resources.csv', nrows=50000)
project_data.shape
```

Out[3]:

```
(50000, 17)
```

In [4]:

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

Out[4]:

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

In [5]:

```
resource_data.head(2)
```

Out[5]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [6]:

```
print(project_data.shape)
print(resource_data.shape)
```

```
(50000, 17)
(50000, 4)
```

In [7]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[7]:

	id	price	quantity
0	p000027	782.13	15
1	p000052	114.98	2

In [8]:

```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
project_data.shape
```

Out[8]:

```
(50000, 19)
```

preprocessing of project_subject_categories

In [9]:

```
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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&
", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'T
he')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Scie
nce"
        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
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]))
```

preprocessing of project_subject_subcategories

In [10]:

```
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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&
", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'T
he')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Scie
nce"
        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]))
```

Text preprocessing (Project_essay)

In [11]:

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

In [12]:

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

Out[12]:

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

In [13]:

```
# printing some random essays.
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[2000])
print("="*50)
print(project_data['essay'].values[4999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. We
are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language t
o our school. \r\n\r\n We have over 24 languages represented in our English Learner program with stu
dents at every level of mastery.  We also have over 40 countries represented with the families withi
n 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 of your world.\"-Lud
wig Wittgenstein  Our English learner's have a strong support system at home that begs for more reso
urces.  Many times our parents are learning to read and speak English along side of their children.
Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter r
ecognition, 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 home is able to assist.  All fam
ilies with students within the Level 1 proficiency status, will be a offered to be a part of this pr
ogram.  These educational videos will be specially chosen by the English 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 educational dvd's for the years to come fo
r other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at lea
st most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of th
e 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to ge
t together and celebrate. Around Halloween there is a whole school parade to show off the beautiful
costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the stude
nts, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard wor
k put in during the school year, with a dunk tank being the most popular activity.My students will u
se these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I w
ill only have a total of ten in the classroom and not enough for each student to have an individual
one, they will be used in a variety of ways. During independent reading time they will be used as sp
ecial 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 used by the students wh
o need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhene
ver asked what the classroom is missing, my students always say more Hokki Stools. They can't get th
eir fill of the 5 stools we already have. When the students are sitting in group with me on the Hokk
i 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 taken. There are always students who h
ead over to the kidney table to get one of the stools who are disappointed as there are not enough o
f them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compro
mise that allow my students to do desk work and move at the same time. These stools will help studen
ts to meet their 60 minutes a day of movement by allowing them to activate their core muscles for ba
lance while they sit. For many of my students, these chairs will take away the barrier that exists i
n 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 d
esks, and a teacher in front of the room? A typical day in our room is nothing like that. I work har
d to create a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nM
y class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey atten
d 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 sep
arating the classrooms. These 9 and 10 year-old students are very eager learners; they are like spon
ges, absorbing all the information and experiences and keep on wanting more.With these resources suc
h 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 environ
ment. Creating a classroom environment is very important in the success in each and every child's ed
ucation. 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 the
m developed, and then hung in our classroom ready for their first day of 4th grade.  This kind gestu
re 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 fro
m day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom read
y. Please consider helping with this project to make our new school year a very successful one. Than
k you!nannan
==================================================
Describing my students isn't an easy task.  Many would say that they are inspirational, creative, an
d hard-working.  They are all unique - unique in their interests, their learning, their abilities, a
nd so much more.  What they all have in common is their desire to learn each day, despite difficulti
es that they encounter.  \r\nOur classroom is amazing - because we understand that everyone learns a
t their own pace.  As the teacher, I pride myself in making sure my students are always engaged, mot
ivated, and inspired to create their own learning! \r\nThis project is to help my students choose se
```

```
ating that is more appropriate for them, developmentally.  Many students tire of sitting in chairs d
uring lessons, and having different seats available helps to keep them engaged and learning.\r\nFlex
ible seating is important in our classroom, as many of our students struggle with attention, focus,
and engagement.  We currently have stability balls for seating, as well as regular chairs, but these
stools will help students who have trouble with balance, or find it difficult to sit on a stability
ball for a long period of time.  We are excited to try these stools as a part of our engaging classr
oom community!nannan
==================================================
Loud and proud are who we are.  We are a special basketball family like no other.  Our school is in
a great community with vast diverseness. We are surrounded by colleges and low income housing.  We p
ride ourselves in preparing our athletes to be great on and off the court.\r\n\r\nOur students recit
e every day that, \"We are destined for greatness.\"  I believe this wholeheartedly.  I am forming w
inners in life and in basketball.  A great of kids is coming your way!We need socks to add to our tw
o uniforms.  Every basketball season our girls basketball team strives to play their best.  Not only
do I push them to give it all on the court I also to teach them to take pride in how they look on th
e team.  We want to look like a team from head to toe.\r\n\r\nGirls should feel good about themselve
s as they play ball and look good on and off the court.  I have seen lime green socks, purple socks,
and all the crazy mismatched socks there is.  We need uniformity all the way around.nannan
==================================================
```

In [14]:

```
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 [15]:

```
sent = decontracted(project_data['essay'].values[2000])
print(sent)# \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)
print("="*50)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative, a
nd hard-working.  They are all unique - unique in their interests, their learning, their abilities,
and so much more.  What they all have in common is their desire to learn each day, despite difficult
ies that they encounter.  \r\nOur classroom is amazing - because we understand that everyone learns
at their own pace.  As the teacher, I pride myself in making sure my students are always engaged, mo
tivated, and inspired to create their own learning! \r\nThis project is to help my students choose s
eating that is more appropriate for them, developmentally.  Many students tire of sitting in chairs
during lessons, and having different seats available helps to keep them engaged and learning.\r\nFle
xible seating is important in our classroom, as many of our students struggle with attention, focus,
and engagement.  We currently have stability balls for seating, as well as regular chairs, but these
stools will help students who have trouble with balance, or find it difficult to sit on a stability
ball for a long period of time.  We are excited to try these stools as a part of our engaging classr
oom community!nannan
Describing my students is not an easy task.  Many would say that they are inspirational, creative, a
nd hard-working.  They are all unique - unique in their interests, their learning, their abilities,
and so much more.  What they all have in common is their desire to learn each day, despite difficult
ies that they encounter.    Our classroom is amazing - because we understand that everyone learns at
their own pace.  As the teacher, I pride myself in making sure my students are always engaged, motiv
ated, and inspired to create their own learning!   This project is to help my students choose seatin
g that is more appropriate for them, developmentally.  Many students tire of sitting in chairs durin
g lessons, and having different seats available helps to keep them engaged and learning.  Flexible s
eating is important in our classroom, as many of our students struggle with attention, focus, and en
gagement.  We currently have stability balls for seating, as well as regular chairs, but these stool
s will help students who have trouble with balance, or find it difficult to sit on a stability ball
for a long period of time.  We are excited to try these stools as a part of our engaging classroom c
ommunity!nannan
==================================================
```

In [16]:

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

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative, a
nd hard-working.  They are all unique - unique in their interests, their learning, their abilities,
and so much more.  What they all have in common is their desire to learn each day, despite difficult
ies that they encounter.    Our classroom is amazing - because we understand that everyone learns at
their own pace.  As the teacher, I pride myself in making sure my students are always engaged, motiv
ated, and inspired to create their own learning!   This project is to help my students choose seatin
g that is more appropriate for them, developmentally.  Many students tire of sitting in chairs durin
g lessons, and having different seats available helps to keep them engaged and learning.  Flexible s
eating is important in our classroom, as many of our students struggle with attention, focus, and en
gagement.  We currently have stability balls for seating, as well as regular chairs, but these stool
s will help students who have trouble with balance, or find it difficult to sit on a stability ball
for a long period of time.  We are excited to try these stools as a part of our engaging classroom c
ommunity!nannan
```

In [17]:

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

```
Describing my students is not an easy task Many would say that they are inspirational creative and h
ard working They are all unique unique in their interests their learning their abilities and so much
more What they all have in common is their desire to learn each day despite difficulties that they e
ncounter Our classroom is amazing because we understand that everyone learns at their own pace As th
e teacher I pride myself in making sure my students are always engaged motivated and inspired to cre
ate their own learning This project is to help my students choose seating that is more appropriate f
or them developmentally Many students tire of sitting in chairs during lessons and having different
seats available helps to keep them engaged and learning Flexible seating is important in our classro
om as many of our students struggle with attention focus and engagement We currently have stability
balls for seating as well as regular chairs but these stools will help students who have trouble wit
h balance or find it difficult to sit on a stability ball for a long period of time We are excited t
o try these stools as a part of our engaging classroom community nannan
```

In [18]:

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

In [ ]:

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

```
 68%|██████████████████████▍           | 33970/50000 [01:00<00:21, 744.37it/s]
```

In [20]:

```
# after preprocesing
preprocessed_essays[2000]
```

Out[20]:

```
'describing students not easy task many would say inspirational creative hard working unique unique
interests learning abilities much common desire learn day despite difficulties encounter classroom a
mazing understand everyone learns pace teacher pride making sure students always engaged motivated i
nspired create learning project help students choose seating appropriate developmentally many studen
ts tire sitting chairs lessons different seats available helps keep engaged learning flexible seatin
g important classroom many students struggle attention focus engagement currently stability balls se
ating well regular chairs stools help students trouble balance find difficult sit stability ball lon
g period time excited try stools part engaging classroom community nannan'
```

Adding a new feature to the preprocessed_essays to the project_data

In [21]:

```
##### Adding a new feature to the preprocessed_essays to the project_data for avg w2v and tfidf
project_data['preprocessed_essays']=preprocessed_essays
```

Preprocessing of project_title

In [22]:

```
sent_0=project_data["project_title"].values[11]
print(sent_0)
print("="*50)

sent_1000=project_data["project_title"].values[34]
print(sent_1000)
print("="*50)

sent_1500=project_data["project_title"].values[147]
print(sent_1500)
print("="*50)

sent_1500=project_data["project_title"].values[1277]
print(sent_1500)
print("="*50)
```

```
Elevating Academics and Parent Rapports Through Technology
==================================================
\"Have A Ball!!!\"
==================================================
Who needs a Chromebook?\r\nWE DO!!
==================================================
Time Keeper=Empathy Builder
==================================================
```

In [23]:

```
# 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 [24]:

```
sent = decontracted(project_data['project_title'].values[34])
print(sent)
print("="*50)
```

```
\"Have A Ball!!!\"
==================================================
```

In [25]:

```
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
 Have A Ball!!!
```

In [26]:

```
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
 Have A Ball
```

In [27]:

```
from tqdm import tqdm
preprocessed_title = []
# 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 not in stopwords)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|████████████████████████████████| 50000/50000 [00:03<00:00, 16227.57it/s]
```

In [28]:

```
preprocessed_title[34]
```

Out[28]:

```
'have a ball'
```

Adding a new feature to the project_title

In [29]:

```
##### Adding a new feature to the preprocessed_title to the project_data for avg w2v and tfidf w2v
project_data['preprocessed_title']=preprocessed_title
```

In [30]:

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

Out[30]:

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

2 rows × 22 columns

In [31]:

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

```
(50000, 21)
(50000,)
```

Out[31]:

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

1 rows × 21 columns

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

In [32]:

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

In [33]:

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

```
(22445, 21) (22445,)
(11055, 21) (11055,)
(16500, 21) (16500,)
```

Preparing Data For Models

Make Data Model Ready: encoding numerical, categorical features

Vectorizing categorical data

In [34]:

```
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['clean_categories'].values)
vectorizer.fit(X_cv['clean_categories'].values)
vectorizer.fit(X_test['clean_categories'].values)
print(vectorizer.get_feature_names())


categories_one_hot_train = vectorizer.transform(X_train['clean_categories'].values)
categories_one_hot_cv = vectorizer.transform(X_cv['clean_categories'].values)
categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].values)
print("Shape of matrix after one hot encoding ",categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",categories_one_hot_cv.shape)
print("Shape of matrix after one hot encoding ",categories_one_hot_test.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health
_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding  (22445, 9)
Shape of matrix after one hot encoding  (11055, 9)
Shape of matrix after one hot encoding  (16500, 9)
```

In [35]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values)
vectorizer.fit(X_cv['clean_subcategories'].values)
vectorizer.fit(X_test['clean_subcategories'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot_train = vectorizer.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)
print("Shape of matrix after one hot encoding ",sub_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",sub_categories_one_hot_cv.shape)
print("Shape of matrix after one hot encoding ",sub_categories_one_hot_test.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civi
cs_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences',
'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'Histo
ry_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience
', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathem
atics', 'Literacy']
Shape of matrix after one hot encoding  (22445, 30)
Shape of matrix after one hot encoding  (11055, 30)
Shape of matrix after one hot encoding  (16500, 30)
```

In [36]:

```
#One Hot Encode - School States
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [37]:

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

In [38]:

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

school_state_categories_one_hot_train = vectorizer.transform(X_train['school_state'].values)
school_state_categories_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)
school_state_categories_one_hot_test = vectorizer.transform(X_test['school_state'].values)
print("Shape of matrix after one hot encoding ",school_state_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",school_state_categories_one_hot_cv.shape)
print("Shape of matrix after one hot encoding ",school_state_categories_one_hot_test.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'NH', 'SD', 'NE', 'AK', 'DE', 'WV', 'ME', 'NM', 'HI', 'DC', 'KS', 'ID
', 'IA', 'AR', 'CO', 'MN', 'OR', 'MS', 'KY', 'NV', 'MD', 'CT', 'TN', 'AL', 'UT', 'WI', 'VA', 'AZ', '
NJ', 'OK', 'MA', 'LA', 'WA', 'MO', 'IN', 'OH', 'PA', 'MI', 'GA', 'SC', 'IL', 'NC', 'FL', 'TX', 'NY',
'CA']
Shape of matrix after one hot encoding  (22445, 51)
Shape of matrix after one hot encoding  (11055, 51)
Shape of matrix after one hot encoding  (16500, 51)
```

In [39]:

```
#One Hot Encode - Project Grade Category
my_counter = Counter()
for project_grade in project_data['project_grade_category'].values:
    my_counter.update(project_grade.split())
```

In [40]:

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

In [41]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values)
vectorizer.fit(X_cv['project_grade_category'].values)
vectorizer.fit(X_test['project_grade_category'].values)
print(vectorizer.get_feature_names())

project_grade_categories_one_hot_train = vectorizer.transform(X_train['project_grade_category'].values)
project_grade_categories_one_hot_cv = vectorizer.transform(X_cv['project_grade_category'].values)
project_grade_categories_one_hot_test = vectorizer.transform(X_test['project_grade_category'].values)
print("Shape of matrix after one hot encoding ",project_grade_categories_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",project_grade_categories_one_hot_cv.shape)
print("Shape of matrix after one hot encoding ",project_grade_categories_one_hot_test.shape)
```

```
['9-12', '6-8', '3-5', 'PreK-2', 'Grades']
Shape of matrix after one hot encoding  (22445, 5)
Shape of matrix after one hot encoding  (11055, 5)
Shape of matrix after one hot encoding  (16500, 5)
```

In [42]:

```
#one hot encode teacher prefix
my_counter = Counter()
for teacher_prefix in project_data['teacher_prefix'].values:
    teacher_prefix = str(teacher_prefix)
    my_counter.update(teacher_prefix.split())
```

In [43]:

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

In [44]:

```
## https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is-an-invalid-document

vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['teacher_prefix'].values.astype("U"))
vectorizer.fit(X_cv['teacher_prefix'].values.astype("U"))
vectorizer.fit(X_test['teacher_prefix'].values.astype("U"))
print(vectorizer.get_feature_names())

teacher_prefix_categories_one_hot_train = vectorizer.transform(X_train['teacher_prefix'].values.astype("U"))
teacher_prefix_categories_one_hot_cv = vectorizer.transform(X_cv['teacher_prefix'].values.astype("U"))
teacher_prefix_categories_one_hot_test = vectorizer.transform(X_test['teacher_prefix'].values.astype("U"))

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

```
['nan', 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encoding  (22445, 6)
Shape of matrix after one hot encoding  (11055, 6)
Shape of matrix after one hot encoding  (16500, 6)
```

Vectorizing Text data

Bag of words on essays

In [45]:

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

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


print('Bow on essay')
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)

print('-'*50)
```

```
Bow on essay
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
--------------------------------------------------
```

Bag of words on project title

In [46]:

```
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['project_title'].values)
X_train_title_bow = vectorizer.transform(X_train['project_title'].values)
X_cv_title_bow = vectorizer.transform(X_cv['project_title'].values)
X_test_title_bow= vectorizer.transform(X_test['project_title'].values)

print('Bow on project title')
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
print('='*50)
```

```
Bow on project title
(22445, 2692) (22445,)
(11055, 2692) (11055,)
(16500, 2692) (16500,)
==================================================
```

TFIDF vectorizer on essays

In [47]:

```
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'].values)# fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].values)
X_cv_essay_tfidf = vectorizer.transform(X_cv['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].values)

print('Tfidf vectrizer on essay')
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)

print('-'*50)
```

```
Tfidf vectrizer on essay
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
--------------------------------------------------
```

TFIDF vectorizer on project title

In [48]:

```
vectorizer.fit(X_train['project_title'].values)
X_train_title_tfidf = vectorizer.transform(X_train['project_title'].values)
X_cv_title_tfidf = vectorizer.transform(X_cv['project_title'].values)
X_test_title_tfidf = vectorizer.transform(X_test['project_title'].values)

print('tfidf vectorizer on project title')
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print('='*50)
```

```
tfidf vectorizer on project title
(22445, 2692) (22445,)
(11055, 2692) (11055,)
(16500, 2692) (16500,)
==================================================
```

Vectorizing Numerical features

Vectorizing- teacher number of previously posted projects

In [49]:

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

X_train_tnopp_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(
-1,1))
X_cv_tnopp_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_tnopp_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1
,1))

print("After vectorizations")
print(X_train_tnopp_norm.shape, y_train.shape)
print(X_cv_tnopp_norm.shape, y_cv.shape)
print(X_test_tnopp_norm.shape, y_test.shape)
print("="*100)
```

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

Vectorizing - price

In [50]:

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


#https://datascience.stackexchange.com/questions/11928/valueerror-input-contains-nan-infinity-or-a-value-too-large-for-dtypefloat32
X_train['price'].fillna(X_train['price'].mean(), inplace=True)
X_cv['price'].fillna(X_cv['price'].mean(), inplace=True)
X_test['price'].fillna(X_test['price'].mean(), inplace=True)
normalizer.fit(X_train['price'].values.reshape(1,-1))

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

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

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

Vectorizing quantity

In [51]:

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



X_train_quantity_norm = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_quantity_norm = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quantity_norm = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

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

Concatinating all the features

categorical features

In [52]:

```
# merging all the categorical features
from scipy.sparse import hstack
categorical_tr=hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_categories_one_hot_trai
n,project_grade_categories_one_hot_train,teacher_prefix_categories_one_hot_train ))
categorical_cv=hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_categories_one_hot_cv,project
_grade_categories_one_hot_cv,teacher_prefix_categories_one_hot_cv ))
categorical_test=hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_categories_one_hot_test
,project_grade_categories_one_hot_test,teacher_prefix_categories_one_hot_test))

print('='*50)

print('final datamatrix')
print(categorical_tr.shape,  y_train.shape)
print(categorical_cv.shape,  y_cv.shape)
print(categorical_test.shape, y_test.shape)
```

```
==================================================
final datamatrix
(22445, 101) (22445,)
(11055, 101) (11055,)
(16500, 101) (16500,)
```

numerical features

In [53]:

```
# merging all the numerical features
import scipy as sp
numerical_tr=sp.hstack((X_train_tnopp_norm,X_train_price_norm,X_train_quantity_norm))
numerical_cv=sp.hstack((X_cv_tnopp_norm,X_cv_price_norm,X_cv_quantity_norm))
numerical_test=sp.hstack((X_test_tnopp_norm,X_test_price_norm,X_test_quantity_norm))

print('='*100)

print('final matrix')
print(numerical_tr.shape, y_train.shape)
print(numerical_cv.shape, y_cv.shape)
print(numerical_test.shape, y_test.shape)
```

```
====================================================================================================
final matrix
(22445, 3) (22445,)
(11055, 3) (11055,)
(16500, 3) (16500,)
```

Applying Knn on categorical+numerical features + project_title(BOW) + preprocessed_essay (BOW)

In [54]:

```
#  creating the matrix
x_tr=hstack((categorical_tr,numerical_tr,X_train_essay_bow,X_train_title_bow)).tocsr()
x_cv=hstack((categorical_cv,numerical_cv,X_cv_essay_bow,X_cv_title_bow)).tocsr()
x_test=hstack((categorical_test,numerical_test,X_test_essay_bow,X_test_title_bow)).tocsr()

print('final matrix')
print(x_tr.shape,   y_train.shape)
print(x_cv.shape,   y_cv.shape)
print(x_test.shape, y_test.shape)
```

```
final matrix
(22445, 7796) (22445,)
(11055, 7796) (11055,)
(16500, 7796) (16500,)
```

Hyper parameter Tuning

In [55]:

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

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

    return y_data_pred
```

In [56]:

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

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

"""

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

    y_train_pred = batch_predict(neigh, x_tr)
    y_cv_pred = batch_predict(neigh, x_cv)

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

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

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

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

```
  0%|                                                    | 0/5 [00:00<?, ?it/s]
```

```
---------------------------------------------------------------------------
PermissionError                           Traceback (most recent call last)
<ipython-input-56-23b286b91bb4> in <module>
     20     neigh.fit(x_tr, y_train)
     21
---> 22     y_train_pred = batch_predict(neigh, x_tr)
     23     y_cv_pred = batch_predict(neigh, x_cv)
     24

<ipython-input-55-70cbcf3d12d8> in batch_predict(clf, data)
      8     # in this for loop we will iterate unti the last 1000 multiplier
      9     for i in range(0, tr_loop, 1000):
---> 10         y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
     11     # we will be predicting for the last data points
     12     if data.shape[0]%1000 !=0:

~\Anaconda3\lib\site-packages\sklearn\neighbors\classification.py in predict_proba(self, X)
    191         X = check_array(X, accept_sparse='csr')
```

```
    192
--> 193         neigh_dist, neigh_ind = self.kneighbors(X)
    194
    195         classes_ = self.classes_

~\Anaconda3\lib\site-packages\sklearn\neighbors\base.py in kneighbors(self, X, n_neighbors, return_d
istance)
    433                 X, self._fit_X, reduce_func=reduce_func,
    434                 metric=self.effective_metric_, n_jobs=n_jobs,
--> 435                 **kwds))
    436
    437         elif self._fit_method in ['ball_tree', 'kd_tree']:

~\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py in pairwise_distances_chunked(X, Y, reduce
_func, metric, n_jobs, working_memory, **kwds)
   1278             X_chunk = X[sl]
   1279         D_chunk = pairwise_distances(X_chunk, Y, metric=metric,
-> 1280                                      n_jobs=n_jobs, **kwds)
   1281         if ((X is Y or Y is None)
   1282                 and PAIRWISE_DISTANCE_FUNCTIONS.get(metric, None)

~\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py in pairwise_distances(X, Y, metric, n_jobs
, **kwds)
   1404         func = partial(distance.cdist, metric=metric, **kwds)
   1405
-> 1406     return _parallel_pairwise(X, Y, func, n_jobs, **kwds)
   1407
   1408

~\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py in _parallel_pairwise(X, Y, func, n_jobs,
**kwds)
   1071     ret = Parallel(n_jobs=n_jobs, verbose=0)(
   1072         fd(X, Y[s], **kwds)
-> 1073         for s in gen_even_slices(_num_samples(Y), effective_n_jobs(n_jobs)))
   1074
   1075     return np.hstack(ret)

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in __call__(self, iterable)
    938                 self._backend.stop_call()
    939             if not self._managed_backend:
--> 940                 self._terminate_backend()
    941             self._jobs = list()
    942             self._pickle_cache = None

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in _terminate_backend(self)
    694     def _terminate_backend(self):
    695         if self._backend is not None:
--> 696             self._backend.terminate()
    697
    698     def _dispatch(self, batch):

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py in terminate(self)
    528             # in latter calls but we free as much memory as we can by deleting
    529             # the shared memory
--> 530             delete_folder(self._workers._temp_folder)
    531             self._workers = None
    532

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\disk.py in delete_folder(folder_path, onerror
)
    113             while True:
    114                 try:
--> 115                     shutil.rmtree(folder_path, False, None)
    116                     break
    117                 except (OSError, WindowsError):

~\Anaconda3\lib\shutil.py in rmtree(path, ignore_errors, onerror)
    492             os.close(fd)
    493     else:
--> 494         return _rmtree_unsafe(path, onerror)
    495
    496 # Allow introspection of whether or not the hardening against symlink

~\Anaconda3\lib\shutil.py in _rmtree_unsafe(path, onerror)
    387                 os.unlink(fullname)
    388             except OSError:
--> 389                 onerror(os.unlink, fullname, sys.exc_info())
    390     try:
    391         os.rmdir(path)

~\Anaconda3\lib\shutil.py in _rmtree_unsafe(path, onerror)
    385         else:
    386             try:
```

```
--> 387                 os.unlink(fullname)
    388             except OSError:
    389                 onerror(os.unlink, fullname, sys.exc_info())

PermissionError: [WinError 32] The process cannot access the file because it is being used by anothe
r process: 'C:\\Users\\user\\AppData\\Local\\Temp\\joblib_memmapping_folder_3920_2211802700\\3920-13
97153976-2787517268e0455a9c1833081de6846c.pkl'
```

grid search cv

In [57]:

```
from sklearn.model_selection import GridSearchCV
def perform_grid_search(X_tr, y_tr, cv_value, title):
    # Our knn model
    knn = KNeighborsClassifier()

    neighbors =  [3, 15, 25, 51, 101]

    parameters = {'n_neighbors':neighbors}

    # https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
    # Set n_jobs = -1 to use all the processors.

    # Increasing the value of cv parameter results in getting more robust value.
    clf = GridSearchCV(knn, parameters, cv=cv_value, scoring='roc_auc', return_train_score=True,n_jobs=-1)
    clf.fit(X_tr, y_tr)

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

    plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')

    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           train_auc - train_auc_std,train_auc + train_auc_std,
                           alpha=0.2,color='darkblue')

    plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           cv_auc - cv_auc_std,cv_auc + cv_auc_std,
                           alpha=0.2,color='darkorange')

    plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
    plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')

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


    # I return clf in order to get the different values like:
    # - best_score_
    # - best_params_
    # - best_estimator_
    return clf
```

In [58]:

```
plot_and_clf = perform_grid_search(x_tr, y_train,
                                   10, "Hyper parameter Vs Auc")
```

```
---------------------------------------------------------------------------
_RemoteTraceback                          Traceback (most recent call last)
_RemoteTraceback:
"""
Traceback (most recent call last):
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\metrics\scorer.py", line 182, in __call__
    y_pred = clf.decision_function(X)
AttributeError: 'KNeighborsClassifier' object has no attribute 'decision_function'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\externals\joblib\externals\loky\process_ex
ecutor.py", line 418, in _process_worker
```

```
    r = call_item()
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\externals\joblib\externals\loky\process_ex
ecutor.py", line 272, in __call__
    return self.fn(*self.args, **self.kwargs)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py", l
ine 567, in __call__
    return self.func(*args, **kwargs)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, i
n __call__
    for func, args, kwargs in self.items]
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py", line 225, i
n <listcomp>
    for func, args, kwargs in self.items]
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 572,
in _fit_and_score
    is_multimetric)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 605,
in _score
    return _multimetric_score(estimator, X_test, y_test, scorer)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 635,
in _multimetric_score
    score = scorer(estimator, X_test, y_test)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\metrics\scorer.py", line 189, in __call__
    y_pred = clf.predict_proba(X)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\neighbors\classification.py", line 193, in
predict_proba
    neigh_dist, neigh_ind = self.kneighbors(X)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\neighbors\base.py", line 435, in kneighbor
s
    **kwds))
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py", line 1280, in pairwi
se_distances_chunked
    n_jobs=n_jobs, **kwds)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py", line 1406, in pairwi
se_distances
    return _parallel_pairwise(X, Y, func, n_jobs, **kwds)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py", line 1067, in _paral
lel_pairwise
    return func(X, Y, **kwds)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\metrics\pairwise.py", line 247, in euclide
an_distances
    distances = safe_sparse_dot(X, Y.T, dense_output=True)
  File "C:\Users\user\Anaconda3\lib\site-packages\sklearn\utils\extmath.py", line 170, in safe_spars
e_dot
    ret = ret.toarray()
  File "C:\Users\user\Anaconda3\lib\site-packages\scipy\sparse\compressed.py", line 1025, in toarray
    out = self._process_toarray_args(order, out)
  File "C:\Users\user\Anaconda3\lib\site-packages\scipy\sparse\base.py", line 1189, in _process_toar
ray_args
    return np.zeros(self.shape, dtype=self.dtype, order=order)
MemoryError: Unable to allocate 1.00 GiB for an array with shape (6644, 20199) and data type float64
"""

The above exception was the direct cause of the following exception:

MemoryError                               Traceback (most recent call last)
<ipython-input-58-8447c2384308> in <module>
      1 plot_and_clf = perform_grid_search(x_tr, y_train,
----> 2                                      10, "Hyper parameter Vs Auc")

<ipython-input-57-ba054e87a1bb> in perform_grid_search(X_tr, y_tr, cv_value, title)
     13     # Increasing the value of cv parameter results in getting more robust value.
     14     clf = GridSearchCV(knn, parameters, cv=cv_value, scoring='roc_auc', return_train_score=T
rue,n_jobs=-1)
---> 15     clf.fit(X_tr, y_tr)
     16
     17     train_auc= clf.cv_results_['mean_train_score']

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in fit(self, X, y, groups, **fit_pa
rams)
    720                 return results_container[0]
    721
--> 722             self._run_search(evaluate_candidates)
    723
    724         results = results_container[0]

~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in _run_search(self, evaluate_candi
dates)
   1189     def _run_search(self, evaluate_candidates):
   1190         """Search all candidates in param_grid"""
-> 1191         evaluate_candidates(ParameterGrid(self.param_grid))
   1192
   1193
```

```
~\Anaconda3\lib\site-packages\sklearn\model_selection\_search.py in evaluate_candidates(candidate_pa
rams)
    709                                   for parameters, (train, test)
    710                                   in product(candidate_params,
--> 711                                              cv.split(X, y, groups)))
    712
    713                 all_candidate_params.extend(candidate_params)

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in __call__(self, iterable)
    928
    929             with self._backend.retrieval_context():
--> 930                 self.retrieve()
    931             # Make sure that we get a last message telling us we are done
    932             elapsed_time = time.time() - self._start_time

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\parallel.py in retrieve(self)
    831             try:
    832                 if getattr(self._backend, 'supports_timeout', False):
--> 833                     self._output.extend(job.get(timeout=self.timeout))
    834                 else:
    835                     self._output.extend(job.get())

~\Anaconda3\lib\site-packages\sklearn\externals\joblib\_parallel_backends.py in wrap_future_result(f
uture, timeout)
    519         AsyncResults.get from multiprocessing."""
    520         try:
--> 521             return future.result(timeout=timeout)
    522         except LokyTimeoutError:
    523             raise TimeoutError()

~\Anaconda3\lib\concurrent\futures\_base.py in result(self, timeout)
    430                 raise CancelledError()
    431             elif self._state == FINISHED:
--> 432                 return self.__get_result()
    433             else:
    434                 raise TimeoutError()

~\Anaconda3\lib\concurrent\futures\_base.py in __get_result(self)
    382     def __get_result(self):
    383         if self._exception:
--> 384             raise self._exception
    385         else:
    386             return self._result

MemoryError: Unable to allocate 1.00 GiB for an array with shape (6644, 20199) and data type float64
```

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

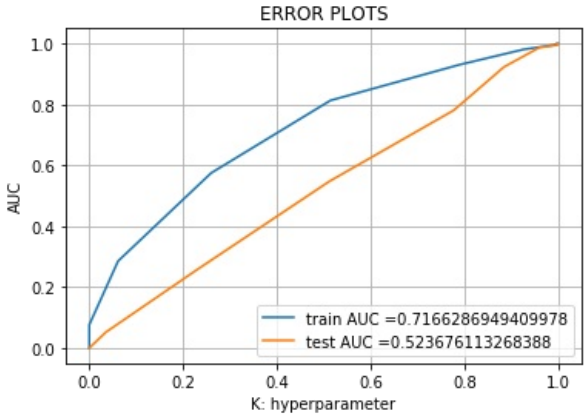
In [57]:

```
best_k = plot_and_clf.best_params_['n_neighbors']
print(best_k)
```

```
15
```

In [58]:

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


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

y_train_pred = batch_predict(neigh, x_tr)
y_test_pred = batch_predict(neigh, x_test)

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

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



Plotting the confusion matrix representation

In [77]:

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

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

confusion matrix for train

In [60]:

```
## TRAIN
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24978625901986937 for threshold 0.8
[[ 166  176]
 [ 355 1547]]
```

In [61]:

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,train_fpr, train_
fpr)), range(2),range(2))
```

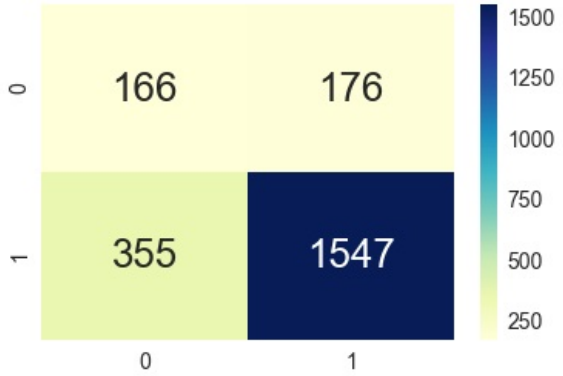
```
the maximum value of tpr*(1-fpr) 0.24978625901986937 for threshold 0.8
```

In [62]:

```
## Heatmaps -> https://likegeeks.com/seaborn-heatmap-tutorial/
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 26}, fmt='g',cmap="YlGnBu")
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18bccef0>
```



confusion matrix for test

In [63]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24985827664399093 for threshold 0.867
[[123 129]
 [633 765]]
```

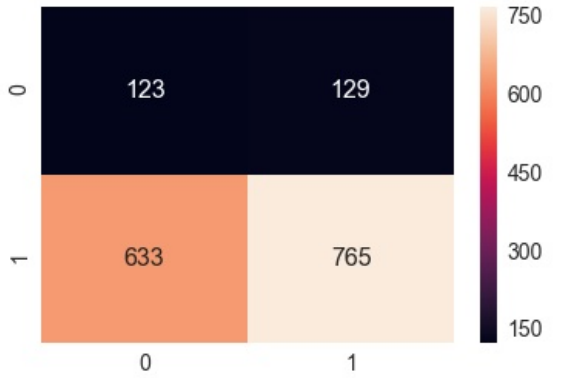
In [64]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)
), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24985827664399093 for threshold 0.867
```

Out[64]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1719e128>
```



In [ ]:

Applying Knn on categorical+numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)

In [65]:

```
x_tr_tfidf=hstack((categorical_tr,numerical_tr,X_train_essay_tfidf,X_train_title_tfidf)).tocsr()
x_cv_tfidf=hstack((categorical_cv,numerical_cv,X_cv_essay_tfidf,X_cv_title_tfidf)).tocsr()
x_test_tfidf=hstack((categorical_test,numerical_test,X_test_essay_tfidf,X_test_title_tfidf)).tocsr()

print('final matrix')
print(x_tr_tfidf.shape,  y_train.shape)
print(x_cv_tfidf.shape,  y_cv.shape)
print(x_test_tfidf.shape, y_test.shape)
print('='*50)
```

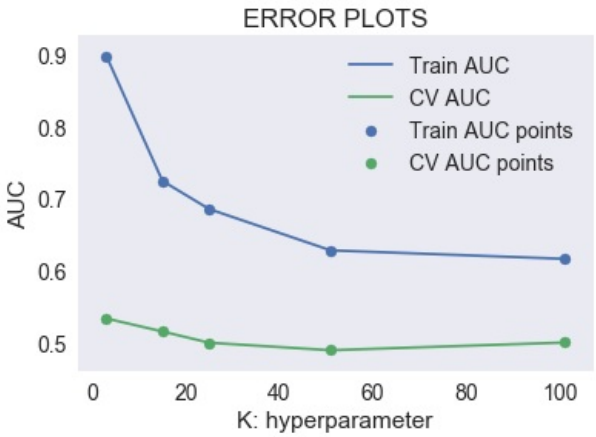
```
final matrix
(2244, 5350) (2244,)
(1106, 5350) (1106,)
(1650, 5350) (1650,)
==================================================
```

Hyper parameter Tuning

In [66]:

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

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

"""

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

    y_train_pred = batch_predict(neigh, x_tr_tfidf)
    y_cv_pred = batch_predict(neigh, x_cv_tfidf)

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

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

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

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

```
100%|██████████████████████████████████████| 5/5 [00:10<00:00,  2.02s/it]
```



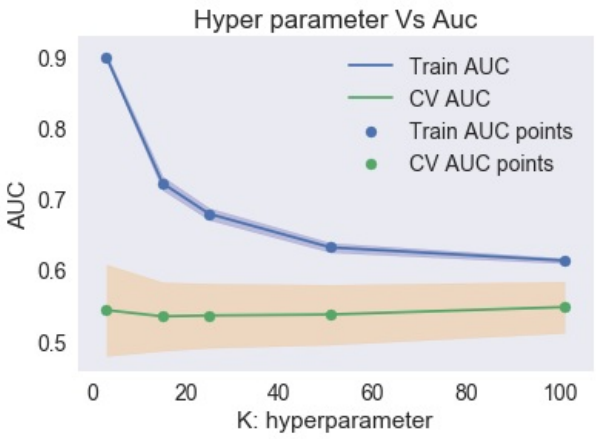
grid search cv

In [67]:

```
from sklearn.model_selection import GridSearchCV
def perform_grid_search(X_tr, y_tr, cv_value, title):
    # Our knn model
    knn = KNeighborsClassifier()


    neighbors = [3, 15, 25, 51, 101]

    parameters = {'n_neighbors':neighbors}

    # https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
    # Set n_jobs = -1 to use all the processors.

    # Increasing the value of cv parameter results in getting more robust value.
    clf = GridSearchCV(knn, parameters, cv=cv_value, scoring='roc_auc', return_train_score=True, n_jobs=-1)
    clf.fit(X_tr, y_tr)

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

    plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')

    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           train_auc - train_auc_std,train_auc + train_auc_std,
                           alpha=0.2,color='darkblue')

    plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           cv_auc - cv_auc_std,cv_auc + cv_auc_std,
                           alpha=0.2,color='darkorange')

    plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
    plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')

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


    # I return clf in order to get the different values like:
    # - best_score_
    # - best_params_
    # - best_estimator_
    return clf
```

In [68]:

```
plot_and_clf = perform_grid_search(x_tr_tfidf, y_train,
                                   10, "Hyper parameter Vs Auc")
```



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

In [69]:

```
best_k = plot_and_clf.best_params_['n_neighbors']
print(best_k)
```

```
101
```

In [70]:

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


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

y_train_pred = batch_predict(neigh, x_tr_tfidf)
y_test_pred = batch_predict(neigh, x_test_tfidf)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

Plotting the confusion matrix representation

confusion matrix for train

In [71]:

```
## TRAIN
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24958106767894395 for threshold 0.842
[[ 178  164]
 [ 678 1224]]
```

In [72]:

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,train_fpr, train_
fpr)), range(2),range(2))
```

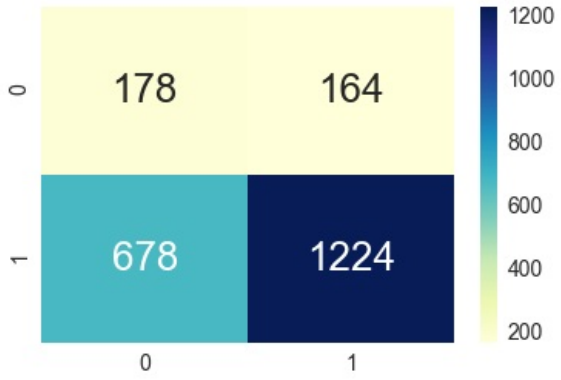
```
the maximum value of tpr*(1-fpr) 0.24958106767894395 for threshold 0.842
```

In [73]:

```
## Heatmaps -> https://likegeeks.com/seaborn-heatmap-tutorial/
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 26}, fmt='g',cmap="YlGnBu")
```

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1436eb38>
```



confusion matrix of test

In [74]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24809460821365587 for threshold 0.851
[[137 115]
 [633 765]]
```

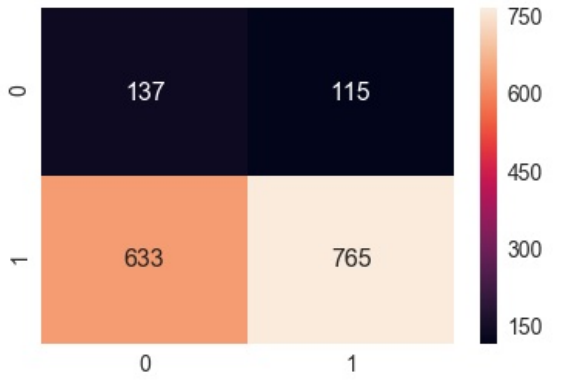
In [75]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)
), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24809460821365587 for threshold 0.851
```

Out[75]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14586860>
```



Applying Knn on categorical+numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)

AVG W2V featurization for essay

In [66]:

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

In [67]:

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

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

```
100%|██████████████████████████████████| 2244/2244 [00:01<00:00, 1805.21it/s]
```

```
2244
300
[ 4.13301348e-02  2.91607272e-02  2.17178913e-03 -1.89942886e-01
  1.10687500e-02  5.23492812e-02 -3.16007359e+00  1.00259304e-01
 -1.61142717e-03  7.51188978e-02 -5.09665870e-02  1.68885709e-02
  1.31457640e-01 -8.15721522e-02 -6.01124630e-02  2.73021533e-02
  5.01968674e-02 -3.64439457e-02  8.12471326e-02 -3.23759239e-03
  2.74928652e-02  4.26374891e-02 -4.54713152e-02 -4.32786707e-02
 -5.16529130e-02 -2.29800391e-02  8.43630870e-02 -4.69340207e-02
 -4.62662533e-02 -1.27822876e-01 -1.97294347e-01 -8.42819022e-02
  5.73517880e-02  1.26587136e-01 -2.53051011e-02 -1.85416489e-02
 -3.80087924e-02 -1.63650728e-02 -6.02556935e-02 -4.38059457e-03
 -1.00475370e-01  9.09865760e-02 -3.09643630e-02 -1.83993191e-01
 -6.87402826e-03 -3.44004359e-02  9.86270109e-03  1.44390511e-02
  3.20967739e-02 -1.32915173e-01 -2.40012739e-02  1.41184783e-03
 -2.23612337e-02  3.33515978e-03  8.48559783e-02 -6.67310728e-02
  8.06440500e-02 -3.96528087e-02 -7.65131815e-02  1.58557646e-01
 -4.31733957e-02 -2.64024674e-02  1.25328371e-01 -5.14658174e-02
 -9.21451348e-02  1.02276616e-01  9.52950907e-02 -7.77681283e-02
  1.19099874e-01 -1.20349587e-01 -9.05281087e-02 -2.38245543e-03
  1.74570853e-02 -5.15124674e-02 -1.15839000e-02 -1.76894026e-01
  2.02618500e-02  2.71200489e-02  3.27860293e-02 -1.72469227e-02
  1.12688343e-01 -2.21615598e-01  5.52767337e-02  1.80129362e-02
 -1.44926220e-01  8.91720000e-03  1.09710848e-01 -1.62347435e-01
  1.76610603e-01 -4.10748696e-03  2.01495724e-02  2.55694891e-03
 -4.76284804e-02  3.75907351e-02 -3.14265402e-02 -2.64996609e-01
 -2.29672411e+00 -7.33718502e-02  1.00549239e-01  1.12421457e-01
 -1.19025713e-01  2.12208043e-02  1.54222413e-01 -9.91902750e-02
  9.11865239e-02  3.12710761e-02  1.41404272e-02 -2.62593533e-01
  3.24806295e-02  3.46180772e-02 -4.49892337e-02  2.71013576e-02
  9.48227677e-02  2.33047809e-01 -2.44985000e-02 -1.74178804e-02
 -2.39609303e-01  5.83000382e-02  4.68569241e-02  6.63043228e-02
  1.23686065e-01  7.23682435e-02 -4.45161204e-02 -1.99030533e-01
  1.00639783e-01  5.19559630e-02  5.32398489e-02  2.59881522e-02
 -4.37769793e-02  1.73611865e-01  7.84132315e-02  4.21212035e-02
 -1.16279130e-02 -1.14633275e-01  5.98597076e-02 -1.56310226e-01
  1.52778978e-01 -8.02732088e-02  7.83780543e-02  2.77907674e-01
  7.80976717e-02 -1.79877467e-02  2.26087935e-02 -2.07727935e-02
 -4.78321978e-03  6.39702174e-04  6.55003674e-02 -3.07091554e-02
  1.80657196e-01 -4.84200685e-02  4.53725946e-02 -5.34336772e-02
  5.30739721e-02  5.83173913e-03  6.24258707e-02  6.30249217e-02
  4.59589272e-02 -1.59888152e-02 -4.25072946e-02  2.66328804e-02
  4.80846728e-02 -2.09153935e-02  9.66012609e-03 -7.85083750e-02
 -1.02715139e-01 -2.07423924e-02 -1.18354390e-01  1.20971777e-01
  7.63788130e-02  4.68108815e-02 -1.10160618e-01 -4.65282391e-02
 -6.06236793e-02 -1.45729948e-01  2.69783392e-02  6.97706641e-02
  8.88464445e-02  1.03215326e-02 -1.50133026e-01 -5.64796489e-02
  4.27977641e-02  3.78300303e-01 -1.06866772e-01 -4.22433924e-02
 -1.05622884e-01 -6.01074457e-02 -1.03412120e-02 -2.71126837e-02
  1.09861838e-01 -1.21860267e-02 -8.85106576e-02 -6.48085022e-02
 -8.89233985e-02 -2.59310772e-02  5.53267098e-02 -1.38583967e-01
 -9.63723261e-02  1.06663818e-01  4.56509171e-02  3.50545482e-02
  2.50717535e-01 -4.19906065e-02 -6.81459203e-02  9.88177500e-02
 -9.38648054e-02  1.07295184e-01  2.84737491e-02 -1.25753042e-01
  8.79459215e-02 -1.55625315e-02 -1.89834217e-02 -4.18840217e-04
  2.51344141e-02 -1.89824020e-01 -1.45276624e-01 -1.28778109e-02
  9.16257391e-03 -5.83539674e-02 -3.69528880e-02 -2.84076957e-03
 -1.57304061e-01 -1.04464470e-01 -9.00299391e-02 -8.53111587e-02
 -1.88618508e+00  1.51812904e-01  3.64492446e-02  3.32645435e-02
 -2.06279815e-02 -1.51787040e-01  4.98285870e-02 -7.95510489e-02
 -4.24318761e-02 -3.90072826e-02 -9.12084696e-02  6.72437478e-02
  4.00739804e-02 -1.30659800e-01  3.60473707e-02  1.36558761e-01
 -6.03358424e-02 -4.68626239e-03 -2.37504815e-01 -4.17742391e-02
 -7.39640978e-02  5.72596685e-02  9.79949565e-03 -7.17485500e-02
  6.24447247e-02 -7.03464457e-02 -5.41788370e-02  1.35749891e-01
  8.12965696e-02 -8.55120370e-02  7.38119361e-02 -3.40834152e-02
  9.41467838e-02 -5.86089783e-03  1.50446413e-01 -6.36669836e-02
 -4.96809022e-02  1.45933913e-02  1.24627065e-02 -1.77270478e-02
  2.10419750e-02 -9.42918130e-02 -9.84316216e-02  5.67672748e-02
  1.17872248e-01  9.96745270e-02 -3.16028457e-02 -2.24673935e-02
 -1.22488673e-01  1.17674698e-01 -8.93057924e-02  5.08069457e-02
  1.21607701e-01 -4.28691630e-02 -2.13509891e-02  1.08424853e-01
  9.81300000e-03  8.58571196e-03 -5.95390358e-02  6.53551587e-02
  2.51323011e-02  2.06126017e-01 -2.13003804e-02 -1.73603261e-03
  3.34507609e-02 -5.78047772e-02  2.88768478e-03 -2.24132424e-02
 -9.36716196e-02 -9.62592935e-02  4.52062407e-02 -1.13300304e-03
 -1.49175902e-02  2.43448270e-01  1.41940667e-01 -3.19332717e-03]
```

In [68]:

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

```
100%|████████████████████████████████████| 1106/1106 [00:00<00:00, 1683.31it/s]
```

In [69]:

```
avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)
```

```
100%|████████████████████████████████████| 1650/1650 [00:00<00:00, 1683.58it/s]
```

AVG W2V featurization for title

In [70]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train_title.append(vector)

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

```
100%|████████████████████████████████████| 2244/2244 [00:00<00:00, 26398.36it/s]
```

```
2244
300
[ 1.98172467e-01  2.44343333e-01 -4.98180000e-01 -3.28640000e-01
  2.34022333e-01 -3.30466667e-02 -2.60626667e+00  5.47761667e-01
 -2.13782667e-01 -6.35530000e-02 -1.56847333e-01 -1.16996667e-01
  4.48406667e-01  2.16460000e-01  3.93182667e-01  1.03872667e-01
 -6.37477667e-02 -3.66480000e-01 -1.18853000e-01  2.33683333e-01
  9.59680000e-02  3.12713333e-02  1.95928000e-01  1.91210000e-01
 -3.17236667e-01 -6.48793333e-02 -2.02221333e-01 -4.51180000e-02
  1.44053133e-01 -1.57322333e-01  7.22310000e-02  1.33339333e-01
 -1.98533333e-01  3.26433333e-02  1.43806667e-01  8.53650000e-02
  2.30633333e-03 -1.20700000e-02  2.22196667e-01  3.03070000e-02
 -4.36476667e-01 -2.12160000e-01  2.23103333e-01  9.83190000e-02
  2.18360000e-01  2.82518333e-01 -1.65966667e-02 -3.16336667e-01
  9.71066667e-02 -2.12626000e-01 -4.56966667e-02  8.14600000e-02
  2.00873333e-01 -1.48006667e-02 -2.60112333e-01  4.58050000e-02
  4.59038000e-01 -7.40626667e-02  1.11176667e-02  1.08120000e-02
  1.94883333e-01  1.97586667e-01 -5.25000000e-02  1.37100333e-01
 -9.29600000e-02  3.72393333e-02  3.78376667e-01  1.59056000e-01
 -2.06504000e-01 -6.48833333e-03 -3.80806667e-01  1.11880000e-01
  3.88846667e-02  2.46428667e-01  1.29763333e-01 -2.62886667e-01
 -3.04969333e-01  2.02591667e-01  1.75361000e-01 -9.07096667e-02
  2.19344333e-01 -6.89000000e-02  1.20156667e-01  4.72571533e-01
 -1.55376333e-01  9.84400000e-03  1.27862333e-01 -6.04233333e-01
  4.52686667e-02 -6.63736000e-02 -1.61920333e-01  1.87113333e-01
 -1.34878333e-01 -5.76920000e-01 -1.21701333e-01 -1.80185000e-01
 -1.88636667e+00 -4.23371667e-01 -4.73333333e-02  6.13356667e-02
  1.05513333e-01  2.22546000e-01  3.74976667e-01 -3.36993333e-01
  1.66203333e-01 -4.02067000e-01  2.66826667e-01  7.93456667e-02
 -1.52613333e-01 -1.60360333e-01 -1.33066667e-01  1.70890000e-01
  8.16068767e-02  1.48982333e-01 -3.35520000e-01 -1.98363333e-01
 -3.59386667e-02 -1.60470000e-01 -5.41933333e-01  1.29831667e-01
 -4.96833333e-02  9.43326667e-02 -4.35503333e-01  1.54715667e-01
  8.21170000e-02  1.01571667e-01  4.51487333e-02  1.37026000e-01
 -2.87747667e-01 -1.01084333e-01  2.05830000e-01  2.54050000e-01
 -1.24666667e-01 -1.40233333e-02  2.60117000e-01 -6.44546667e-01
  3.14709667e-01 -3.13366667e-02 -2.34590000e-01  4.00523000e-01
  1.72743333e-01 -1.28470000e-02  1.48781333e-01  3.31023333e-01
  2.37699333e-01 -4.78923333e-01  2.13273333e-03  5.45100000e-03
  7.75530000e-01 -1.46798667e-01  4.33540000e-02 -1.53513667e-01
  5.75130000e-02 -1.86840000e-01 -1.23485667e-01 -1.01960000e-01
 -1.31356867e-01 -7.80710000e-02 -8.01183333e-02 -4.15600000e-02
 -1.10670000e-01 -6.78356667e-02 -5.72583333e-02 -3.81013333e-01
 -1.92666667e-03 -2.23280000e-01 -6.97136667e-02  2.21400000e-01
  2.96356667e-01  4.06176667e-01 -1.57266667e-02  1.32706667e-01
 -5.65300000e-02  1.25243333e-01  6.26530000e-02  4.22210000e-01
  5.55600000e-02  2.23190000e-02 -1.84510000e-01 -3.18235667e-01
 -2.04717000e-01  3.52300000e-01 -5.28190000e-01 -3.74526667e-02
 -3.66706667e-01  5.21933333e-03 -3.47506667e-02 -3.28893333e-02
  1.85894000e-01 -1.47366667e-02 -1.49039667e-01  2.78571067e-01
  2.48740000e-01 -2.01616667e-02  8.53733333e-02 -9.18946667e-02
 -3.83780000e-01  6.41533333e-02 -2.02166667e-02 -1.32846667e-01
 -1.74416667e-01  1.92582000e-01 -3.80640000e-01  2.80740000e-02
  2.51376000e-01 -2.18387000e-01 -1.46080000e-01  1.05206667e-01
 -3.27150000e-01 -3.91503333e-02 -3.58120000e-01  2.13252667e-01
  6.81333333e-03 -8.50566667e-02 -1.90916667e-01 -4.86860000e-02
  5.79983333e-01  1.53146667e-01 -3.13323333e-01 -1.00300000e-02
 -3.44692000e-01 -1.45974333e-01  5.81332000e-02  1.82780000e-01
 -2.49250000e+00  3.96773333e-01  3.56140000e-01  3.00940000e-01
  7.39137000e-02  8.40763333e-02 -1.13056667e-01 -3.38712000e-01
  1.17216667e-01 -3.05270000e-01  1.17261667e-01  2.74766667e-02
  7.69140000e-02  4.19776667e-01 -2.24290000e-01  1.50495867e-01
  1.01096667e-01  9.47960000e-02 -3.43251333e-01 -8.40033333e-02
 -1.27474333e-01 -1.45770000e-01  3.10129667e-01  5.87263333e-01
  2.67460667e-01  1.86262333e-01  1.62195333e-01  9.70233333e-02
  1.85326667e-01 -7.57450000e-02  3.72187000e-01 -1.97740000e-02
 -2.39023333e-01  6.62596667e-01  6.62927333e-02  6.83276667e-01
  2.77193333e-02 -1.08317667e-01 -7.75066667e-02 -7.56390000e-02
 -2.04473333e-01  1.03680000e-01 -5.92400000e-03  2.68752333e-01
  1.75603333e-02  1.92226667e-01  3.47570000e-01 -1.53549833e-01
 -5.09710000e-02 -1.12270000e-02  7.56666667e-02  2.94400000e-01
 -1.03053333e-01  1.57730000e-01 -1.17185667e-01  3.25830000e-01
  6.17880000e-01  2.37366667e-02 -1.00487600e-01 -1.67586133e-01
  8.88103333e-02  1.05205000e-01 -1.57074667e-01 -2.14975667e-01
  1.56129333e-01  3.22206667e-02 -4.13566667e-02 -1.93788333e-01
  3.89663333e-02  2.17225667e-01 -2.48056667e-01  7.81156667e-02
 -3.85166667e-02  2.41006667e-01 -1.93394000e-01  1.08276667e-01]
```

In [71]:

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

```
100%|████████████████████████████████| 1106/1106 [00:00<00:00, 24576.30it/s]
```

In [72]:

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

```
100%|████████████████████████████████| 1650/1650 [00:00<00:00, 30553.86it/s]
```

merging all the above matrices

In [74]:

```
import scipy as sp
x_tr_avg_w2v=hstack((categorical_tr,numerical_tr,avg_w2v_vectors_train,avg_w2v_vectors_train_title)).tocsr()
x_cv_avg_w2v=hstack((categorical_cv,numerical_cv,avg_w2v_vectors_cv,avg_w2v_vectors_cv_title)).tocsr()
x_test_avg_w2v=hstack((categorical_test,numerical_test,avg_w2v_vectors_test,avg_w2v_vectors_test_title)).tocsr()


print('final matrix')
print(x_tr_avg_w2v.shape,  y_train.shape)
print(x_cv_avg_w2v.shape,  y_cv.shape)
print(x_test_avg_w2v.shape, y_test.shape)
```

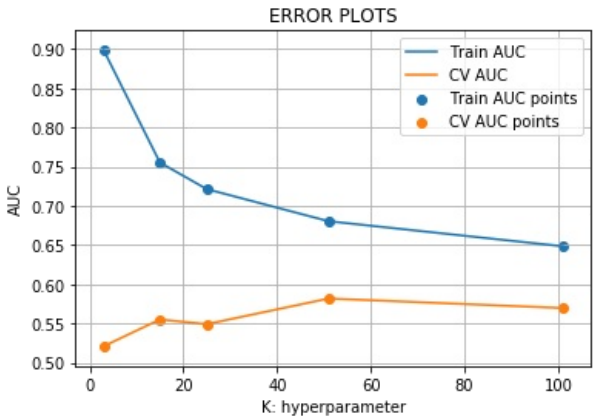
```
final matrix
(2244, 700) (2244,)
(1106, 700) (1106,)
(1650, 700) (1650,)
```

Hyper parameter tuning

In [78]:

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

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

"""

train_auc = []
cv_auc = []
K = [3, 15, 25, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(x_tr_avg_w2v, y_train)

    y_train_pred = batch_predict(neigh, x_tr_avg_w2v)
    y_cv_pred = batch_predict(neigh, x_cv_avg_w2v)

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

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

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

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

```
100%|██████████████████████████████████████████| 5/5 [01:07<00:00, 13.47s/it]
```



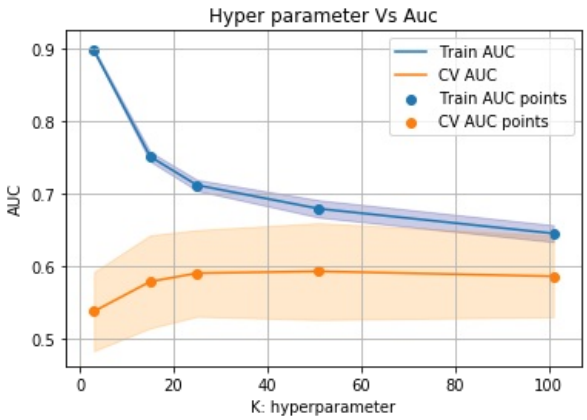
grid search cv

In [79]:

```
from sklearn.model_selection import GridSearchCV
def perform_grid_search(X_tr, y_tr, cv_value, title):
    # Our knn model
    knn = KNeighborsClassifier()

    neighbors = [3, 15, 25, 51, 101]

    parameters = {'n_neighbors':neighbors}

    # https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
    # Set n_jobs = -1 to use all the processors.

    # Increasing the value of cv parameter results in getting more robust value.
    clf = GridSearchCV(knn, parameters, cv=cv_value, scoring='roc_auc', return_train_score=True, n_jobs=-1)
    clf.fit(X_tr, y_tr)

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

    plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')

    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           train_auc - train_auc_std,train_auc + train_auc_std,
                           alpha=0.2,color='darkblue')

    plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           cv_auc - cv_auc_std,cv_auc + cv_auc_std,
                           alpha=0.2,color='darkorange')

    plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
    plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')

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


    # I return clf in order to get the different values like:
    # - best_score_
    # - best_params_
    # - best_estimator_
    return clf
```

In [80]:

```
plot_and_clf = perform_grid_search(x_tr_avg_w2v, y_train,
                                   10, "Hyper parameter Vs Auc")
```



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

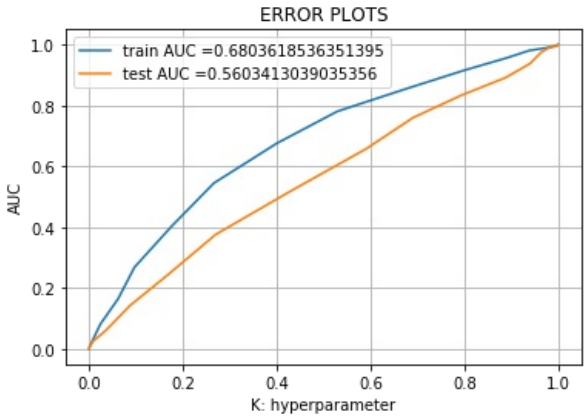
In [81]:

```
best_k = plot_and_clf.best_params_['n_neighbors']
print(best_k)
```

```
51
```

In [82]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(x_tr_avg_w2v, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, x_tr_avg_w2v)
y_test_pred = batch_predict(neigh, x_test_avg_w2v)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



plotting confusion matrix

confusion matrix of Train

In [83]:

```
## TRAIN
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24914503607947744 for threshold 0.824
[[ 161  181]
 [ 417 1485]]
```

In [84]:

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,train_fpr, train_
fpr)), range(2),range(2))
```

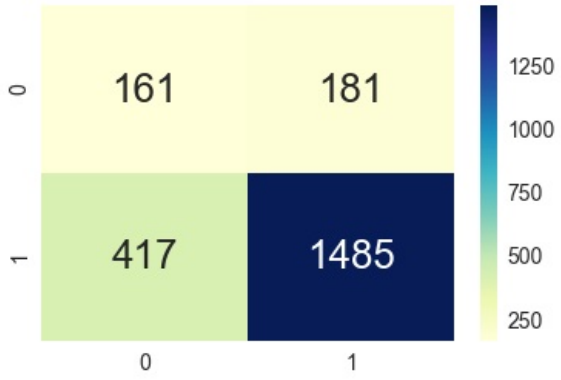
```
the maximum value of tpr*(1-fpr) 0.24914503607947744 for threshold 0.824
```

In [85]:

```
## Heatmaps -> https://likegeeks.com/seaborn-heatmap-tutorial/
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 26}, fmt='g',cmap="YlGnBu")
```

Out[85]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1828b128>
```



confusion matrix of test

In [86]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2443153187200806 for threshold 0.843
[[103 149]
 [478 920]]
```

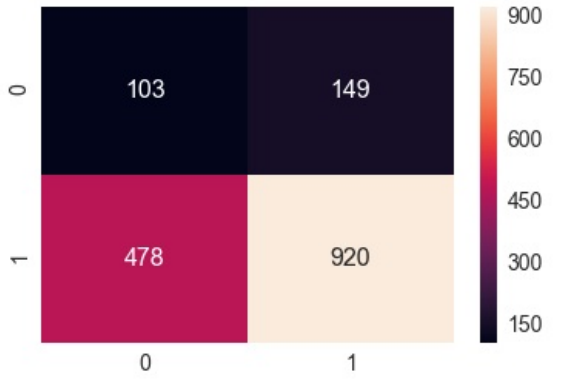
In [87]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)
), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2443153187200806 for threshold 0.843
```

Out[87]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1246bbe0>
```



Applying Knn on categorical+numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

TFIDF weighted W2V on essay

In [88]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_essays'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [89]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_tr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sent
ence.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_tr.append(vector)

print(len(tfidf_w2v_vectors_tr))
print(len(tfidf_w2v_vectors_tr[0]))
```

```
100%|██████████████████████████████████| 2244/2244 [00:08<00:00, 265.08it/s]

2244
300
```

In [90]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sent
ence.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_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

```
100%|██████████████████████████████████| 1106/1106 [00:04<00:00, 227.89it/s]

1106
300
```

In [91]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sent
ence.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_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████| 1650/1650 [00:07<00:00, 214.72it/s]

1650
300
```

TFIDF weighted W2V on title

In [94]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [95]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_tr_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sent
ence.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_tr_title.append(vector)

print(len(tfidf_w2v_vectors_tr_title))
print(len(tfidf_w2v_vectors_tr_title[0]))
```

```
100%|████████████████████████████████████| 2244/2244 [00:00<00:00, 13277.35it/s]

2244
300
```

In [96]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sent
ence.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_cv_title.append(vector)

print(len(tfidf_w2v_vectors_cv_title))
print(len(tfidf_w2v_vectors_cv_title[0]))
```

```
100%|████████████████████████████████| 1106/1106 [00:00<00:00, 13653.58it/s]

1106
300
```

In [97]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sent
ence.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_test_title.append(vector)

print(len(tfidf_w2v_vectors_test_title))
print(len(tfidf_w2v_vectors_test_title[0]))
```

```
100%|████████████████████████████████| 1650/1650 [00:00<00:00, 14347.02it/s]

1650
300
```

merging all the above matrices

In [98]:

```
x_tr_tfidf_w2v=hstack((categorical_tr,numerical_tr,tfidf_w2v_vectors_tr,tfidf_w2v_vectors_tr_title)).tocsr()
x_cv_tfidf_w2v=hstack((categorical_cv,numerical_cv,tfidf_w2v_vectors_cv,tfidf_w2v_vectors_cv_title)).tocsr()
x_test_tfidf_w2v=hstack((categorical_test,numerical_test,tfidf_w2v_vectors_test,tfidf_w2v_vectors_test_title)).to
csr()

print("final matrix")
print(x_tr_tfidf_w2v.shape,   y_train.shape)
print(x_cv_tfidf_w2v.shape,   y_cv.shape)
print(x_test_tfidf_w2v.shape, y_test.shape)
```

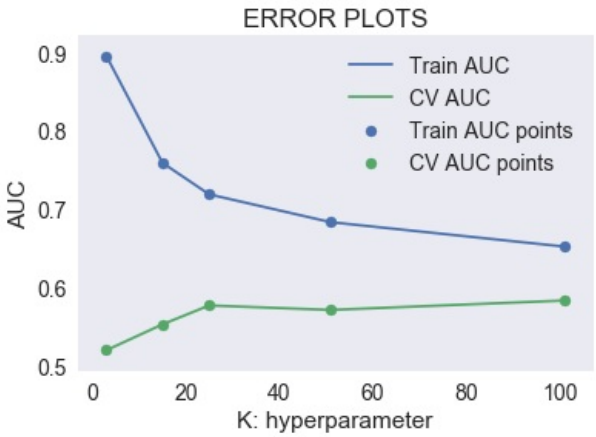
```
final matrix
(2244, 700) (2244,)
(1106, 700) (1106,)
(1650, 700) (1650,)
```

Hyper parameter tuning

In [99]:

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

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

"""

train_auc = []
cv_auc = []
K = [3, 15, 25, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(x_tr_tfidf_w2v, y_train)

    y_train_pred = batch_predict(neigh, x_tr_tfidf_w2v)
    y_cv_pred = batch_predict(neigh, x_cv_tfidf_w2v)

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

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

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

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

```
100%|██████████████████████████████████████| 5/5 [00:57<00:00, 11.53s/it]
```



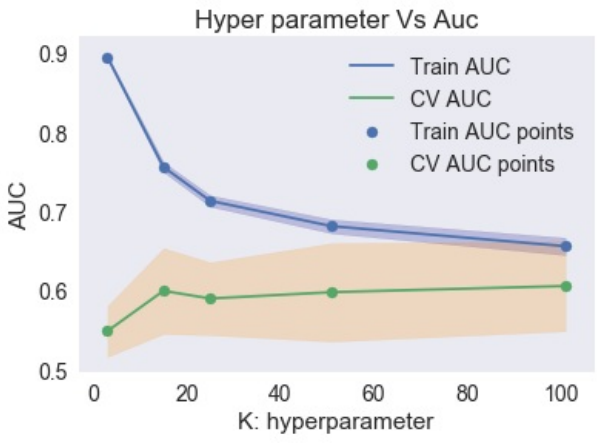
grid search cv

In [100]:

```
from sklearn.model_selection import GridSearchCV
def perform_grid_search(X_tr, y_tr, cv_value, title):
    # Our knn model
    knn = KNeighborsClassifier()

    neighbors =   [3, 15, 25, 51, 101]

    parameters = {'n_neighbors':neighbors}

    # https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
    # Set n_jobs = -1 to use all the processors.

    # Increasing the value of cv parameter results in getting more robust value.
    clf = GridSearchCV(knn, parameters, cv=cv_value, scoring='roc_auc', return_train_score=True, n_jobs=-1)
    clf.fit(X_tr, y_tr)

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

    plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')

    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           train_auc - train_auc_std,train_auc + train_auc_std,
                           alpha=0.2,color='darkblue')

    plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           cv_auc - cv_auc_std,cv_auc + cv_auc_std,
                           alpha=0.2,color='darkorange')

    plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
    plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')

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


    # I return clf in order to get the different values like:
    # - best_score_
    # - best_params_
    # - best_estimator_
    return clf
```

In [101]:

```
plot_and_clf = perform_grid_search(x_tr_tfidf_w2v, y_train,
                                   10, "Hyper parameter Vs Auc")
```



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

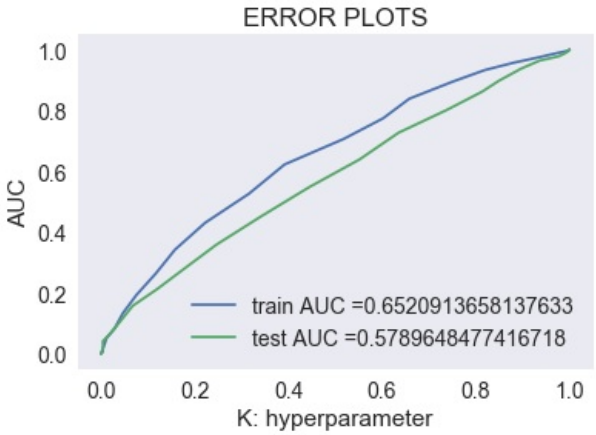
In [102]:

```
best_k = plot_and_clf.best_params_['n_neighbors']
print(best_k)
```

```
101
```

In [103]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(x_tr_tfidf_w2v, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, x_tr_tfidf_w2v)
y_test_pred = batch_predict(neigh, x_test_tfidf_w2v)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



plotting confusion matrix

confusion matrix of train

In [107]:

```
## TRAIN
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24969221298861188 for threshold 0.832
[[ 165  177]
 [ 559 1343]]
```

In [108]:

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,train_fpr, train_
fpr)), range(2),range(2))
```

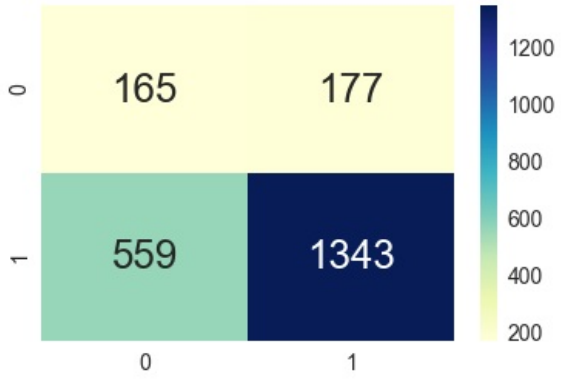
```
the maximum value of tpr*(1-fpr) 0.24969221298861188 for threshold 0.832
```

In [109]:

```
## Heatmaps -> https://likegeeks.com/seaborn-heatmap-tutorial/
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 26}, fmt='g',cmap="YlGnBu")
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18427be0>
```



confusion matrix of test

In [110]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2473387503149408 for threshold 0.832
[[113 139]
 [505 893]]
```

In [111]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)
), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2473387503149408 for threshold 0.832
```

Out[111]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x14ca2a90>
```

Selecting top 2000 features from Set 2 of preprocessed_essay (TFIDF)

In [114]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif
#https://stackoverflow.com/questions/49300193/feature-selection-f-classif-scikit-learn
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.feature_selection import f_classif

## -> https://stackoverflow.com/a/46929321/4433839

selector = SelectKBest(f_classif, k=2000)
selector.fit(X_train_essay_tfidf, y_train)
x_train_tfidf_2000 =selector.transform(X_train_essay_tfidf)
x_cv_tfidf_2000 = selector.transform(X_cv_essay_tfidf)
x_test_tfidf_2000 = selector.transform(X_test_essay_tfidf)
print(x_train_tfidf_2000.shape)
print(x_cv_tfidf_2000.shape)
print(x_test_tfidf_2000.shape)
```

```
(2244, 2000)
(1106, 2000)
(1650, 2000)
```

merging all the matrices

In [115]:

```
x_tr_2000=hstack((categorical_tr,numerical_tr,x_train_tfidf_2000,X_train_title_tfidf)).tocsr()
x_cv_2000=hstack((categorical_cv,numerical_cv,x_cv_tfidf_2000,X_cv_title_tfidf)).tocsr()
x_test_2000=hstack((categorical_test,numerical_test,x_test_tfidf_2000,X_test_title_tfidf)).tocsr()

print('final matrix')
print(x_tr_2000.shape,  y_train.shape)
print(x_cv_2000.shape,  y_cv.shape)
print(x_test_2000.shape, y_test.shape)
print('='*50)
```

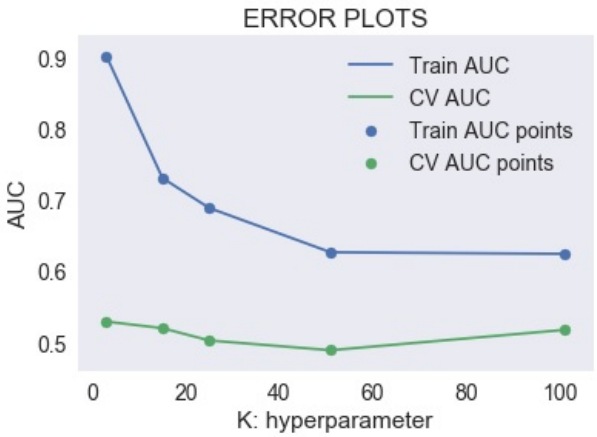
```
final matrix
(2244, 2350) (2244,)
(1106, 2350) (1106,)
(1650, 2350) (1650,)
==================================================
```

Hyper parameter tuning

In [116]:

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

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

"""

train_auc = []
cv_auc = []
K = [3, 15, 25, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(x_tr_2000, y_train)

    y_train_pred = batch_predict(neigh, x_tr_2000)
    y_cv_pred = batch_predict(neigh, x_cv_2000)

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

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

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

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

```
100%|██████████████████████████████████████████| 5/5 [00:07<00:00,  1.43s/it]
```



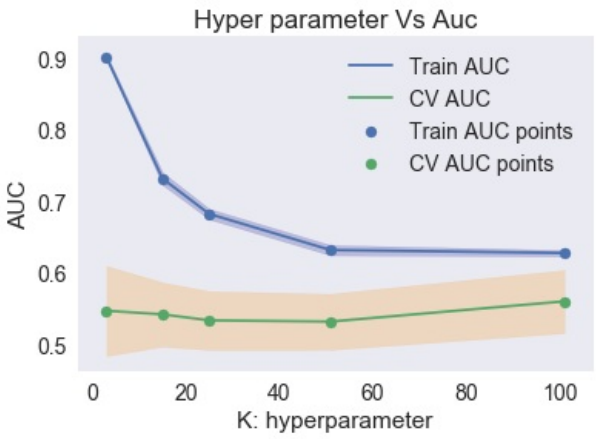
grid search

In [117]:

```
from sklearn.model_selection import GridSearchCV
def perform_grid_search(X_tr, y_tr, cv_value, title):
    # Our knn model
    knn = KNeighborsClassifier()


    neighbors =   [3, 15, 25, 51, 101]

    parameters = {'n_neighbors':neighbors}

    # https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
    # Set n_jobs = -1 to use all the processors.

    # Increasing the value of cv parameter results in getting more robust value.
    clf = GridSearchCV(knn, parameters, cv=cv_value, scoring='roc_auc', return_train_score=True, n_jobs=-1)
    clf.fit(X_tr, y_tr)

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

    plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')

    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           train_auc - train_auc_std,train_auc + train_auc_std,
                           alpha=0.2,color='darkblue')

    plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
    # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
    plt.gca().fill_between(parameters['n_neighbors'],
                           cv_auc - cv_auc_std,cv_auc + cv_auc_std,
                           alpha=0.2,color='darkorange')

    plt.scatter(parameters['n_neighbors'], train_auc, label='Train AUC points')
    plt.scatter(parameters['n_neighbors'], cv_auc, label='CV AUC points')

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


    # I return clf in order to get the different values like:
    # - best_score_
    # - best_params_
    # - best_estimator_
    return clf
```

In [118]:

```
plot_and_clf = perform_grid_search(x_tr_2000, y_train,
                                   10, "Hyper parameter Vs Auc")
```



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

In [120]:

```
best_k = plot_and_clf.best_params_['n_neighbors']
print(best_k)
```

```
101
```

In [121]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(x_tr_2000, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, x_tr_2000)
y_test_pred = batch_predict(neigh, x_test_2000)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

plotting confusion matrix

confusion matrix of train

In [104]:

```
## TRAIN
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24969221298861188 for threshold 0.832
[[ 165  177]
 [ 559 1343]]
```

In [105]:

```
conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,train_fpr, train_
fpr)), range(2),range(2))
```

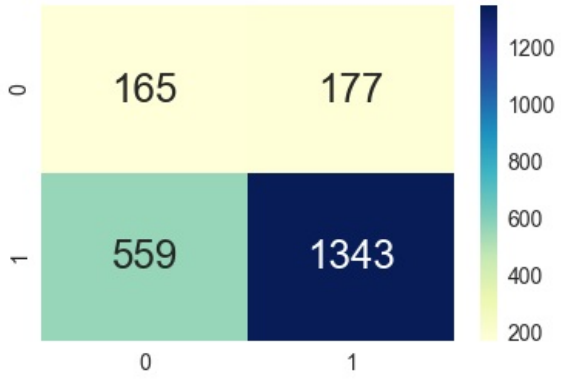
```
the maximum value of tpr*(1-fpr) 0.24969221298861188 for threshold 0.832
```

In [106]:

```
## Heatmaps -> https://likegeeks.com/seaborn-heatmap-tutorial/
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 26}, fmt='g',cmap="YlGnBu")
```

Out[106]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1729b160>
```



confusion matrix on test

In [125]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24974804736709497 for threshold 0.851
[[130 122]
 [590 808]]
```

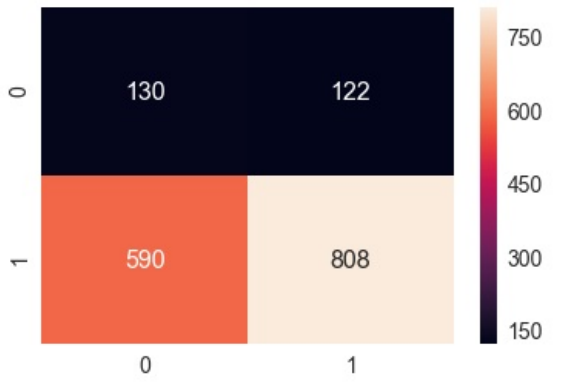
In [126]:

```
conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)
), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24974804736709497 for threshold 0.851
```

Out[126]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18fba9e8>
```



Conclusion

In [ ]:

```
#http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Hyperparameter", "trainAUC", "test AUC" ]
x.add_row(["Bag of Words",15, 0.7166, 0.5263])
x.add_row(["TFIDF", 101, 0.6162, .5550])
x.add_row(["Avgw2v", 51, 0.6803,   0.5603])
x.add_row(["Tfidfw2v", 101, 0.6520, 0.5789])
x.add_row(["Tfidf 2000", 101,0.6240, .541 ])

print(x)
```

In [ ]:

```

```