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	Desc
`project_id`	A unique identifier for the proposed project. **Example:** `p0`
`project_title`	Title of the project. **Exa** • `Art Will Make You H` • `First Grad`
`project_grade_category`	Grade level of students for which the project is targeted. One of the fo enumerated v • `Grades P` • `Grade` • `Grade` • `Grades`

Feature	Desc
project_subject_categories	One or more (comma-separated) subject categories for the project fr following enumerated list of v • Applied Lea • Care & H • Health & S • History & C • Literacy & Lan • Math & Sc • Music & The • Special • W **Exar** • Music & The • Literacy & Language, Math & Sc
school_state	State where school is located (Two-letter U.S. post (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_c **Exampl**
project_subject_subcategories	One or more (comma-separated) subject subcategories for the p **Exar** • Lit • Literature & Writing, Social Sci
project_resource_summary	An explanation of the resources needed for the project. **Exa** • My students need hands on literacy materials to ma sensory needs!<
project_essay_1	First application
project_essay_2	Second application
project_essay_3	Third application
project_essay_4	Fourth application
project_submitted_datetime	Datetime when project application was submitted. **Example:** `2016-( 12:43:5`
teacher_id	A unique identifier for the teacher of the proposed project. **Exa** `bdf8baa8fedef6bfeec7ae4ff1c`
teacher_prefix	Teacher's title. One of the following enumerated v • • • • • • Tea
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same te **Examp**

* 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`

Feature	Description
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

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

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

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

Notes on the Essay Data

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

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_3:** "Close by sharing why your project will make a difference"

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

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [6]:

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

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [7]:

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

In [8]:

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

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

In [9]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns


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


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


project_data.head()
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

In [10]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head()
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[10]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

1.2 preprocessing of `project_subject_categories`

In [11]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# 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", '
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        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]))
```

1.3 preprocessing of `project_subject_subcategories`

In [12]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# 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", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

1.3 Text preprocessing

In [13]:

```
# 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 [14]:

```
project_data.head()
```

Out[14]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

In [15]:

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

In [16]:

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

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom
as well as the STEM journals, which my students really enjoyed.  I would lov
e to implement more of the Lakeshore STEM kits in my classroom for the next
school year as they provide excellent and engaging STEM lessons.My students
come from a variety of backgrounds, including language and socioeconomic sta
tus.  Many of them don't have a lot of experience in science and engineering
and these kits give me the materials to provide these exciting opportunities
for my students.Each month I try to do several science or STEM/STEAM project
s.  I would use the kits and robot to help guide my science instruction in e
ngaging and meaningful ways.  I can adapt the kits to my current language ar
ts pacing guide where we already teach some of the material in the kits like
tall tales (Paul Bunyan) or Johnny Appleseed.  The following units will be t
aught in the next school year where I will implement these kits: magnets, mo
tion, sink vs. float, robots.  I often get to these units and don't know If
I am teaching the right way or using the right materials.    The kits will g
ive me additional ideas, strategies, and lessons to prepare my students in s
cience.It is challenging to develop high quality science activities.  These
kits give me the materials I need to provide my students with science activi
ties that will go along with the curriculum in my classroom.  Although I hav
e some things (like magnets) in my classroom, I don't know how to use them e
ffectively.  The kits will provide me with the right amount of materials and
show me how to use them in an appropriate way.
==================================================
I teach high school English to students with learning and behavioral disabil
ities. My students all vary in their ability level. However, the ultimate go
al is to increase all students literacy levels. This includes their reading,
writing, and communication levels.I teach a really dynamic group of student
s. However, my students face a lot of challenges. My students all live in po
verty and in a dangerous neighborhood. Despite these challenges, I have stud
ents who have the the desire to defeat these challenges. My students all hav
e learning disabilities and currently all are performing below grade level.
My students are visual learners and will benefit from a classroom that fulfi
lls their preferred learning style.The materials I am requesting will allow
my students to be prepared for the classroom with the necessary supplies.  T
oo often I am challenged with students who come to school unprepared for cla
ss due to economic challenges.  I want my students to be able to focus on le
arning and not how they will be able to get school supplies.  The supplies w
ill last all year.  Students will be able to complete written assignments an
d maintain a classroom journal.  The chart paper will be used to make learni
ng more visual in class and to create posters to aid students in their learn
ing.  The students have access to a classroom printer.  The toner will be us
ed to print student work that is completed on the classroom Chromebooks.I wa
nt to try and remove all barriers for the students learning and create oppor
tunities for learning. One of the biggest barriers is the students not havin
g the resources to get pens, paper, and folders. My students will be able to
increase their literacy skills because of this project.
==================================================
\"Life moves pretty fast. If you don't stop and look around once in awhile,
you could miss it.\"  from the movie, Ferris Bueller's Day Off.  Think bac
k...what do you remember about your grandparents?  How amazing would it be t
o be able to flip through a book to see a day in their lives?My second grade
rs are voracious readers! They love to read both fiction and nonfiction book
s.  Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elep
hant, and Mercy Watson. They also love to read about insects, space and plan
ts. My students are hungry bookworms! My students are eager to learn and rea
d about the world around them. My kids love to be at school and are like lit
tle sponges absorbing everything around them. Their parents work long hours
and usually do not see their children. My students are usually cared for by
their grandparents or a family friend. Most of my students do not have someo
ne who speaks English at home. Thus it is difficult for my students to acqui
re language.Now think forward... wouldn't it mean a lot to your kids, nieces
or nephews or grandchildren, to be able to see a day in your life today 30 y
ears from now? Memories are so precious to us and being able to share these
memories with future generations will be a rewarding experience.  As part of
our social studies curriculum, students will be learning about changes over
time.  Students will be studying photos to learn about how their community h
as changed over time.  In particular, we will look at photos to study how th
e land, buildings, clothing, and schools have changed over time.  As a culmi
nating activity, my students will capture a slice of their history and prese
rve it through scrap booking. Key important events in their young lives will
be documented with the date, location, and names.   Students will be using p
hotos from home and from school to create their second grade memories.   The
ir scrap books will preserve their unique stories for future generations to
enjoy.Your donation to this project will provide my second graders with an o
pportunity to learn about social studies in a fun and creative manner.  Thro
ugh their scrapbooks, children will share their story with others and have a
historical document for the rest of their lives.
==================================================
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smalles
t students with the biggest enthusiasm for learning. My students learn in ma
ny different ways using all of our senses and multiple intelligences. I use
a wide range of techniques to help all my students succeed. \r\nStudents in
my class come from a variety of different backgrounds which makes for wonder
ful sharing of experiences and cultures, including Native Americans.\r\nOur
school is a caring community of successful learners which can be seen throug
h collaborative student project based learning in and out of the classroom.
Kindergarteners in my class love to work with hands-on materials and have ma
ny different opportunities to practice a skill before it is mastered. Having
the social skills to work cooperatively with friends is a crucial aspect of
the kindergarten curriculum.Montana is the perfect place to learn about agri
culture and nutrition. My students love to role play in our pretend kitchen
in the early childhood classroom. I have had several kids ask me, \"Can we t
ry cooking with REAL food?\" I will take their idea and create \"Common Core
Cooking Lessons\" where we learn important math and writing concepts while c
ooking delicious healthy food for snack time. My students will have a ground
ed appreciation for the work that went into making the food and knowledge of
where the ingredients came from as well as how it's healthy for their bodie
s. This project would expand our learning of nutrition and agricultural cook
ing recipes by having us peel our own apples to make homemade applesauce, ma
ke our own bread, and mix up healthy plants from our classroom garden in the
spring. We will also create our own cookbooks to be printed and shared with
families. \r\nStudents will gain math and literature skills as well as a lif
e long enjoyment for healthy cooking.nannan
==================================================

My classroom consists of twenty-two amazing sixth graders from different cul
tures and backgrounds. They are a social bunch who enjoy working in partners
and working with groups. They are hard-working and eager to head to middle s
```

```
chool next year. My job is to get them ready to make this transition and mak
e it as smooth as possible. In order to do this, my students need to come to
school every day and feel safe and ready to learn. Because they are getting
ready to head to middle school, I give them lots of choice- choice on where
to sit and work, the order to complete assignments, choice of projects, etc.
Part of the students feeling safe is the ability for them to come into a wel
coming, encouraging environment. My room is colorful and the atmosphere is c
asual. I want them to take ownership of the classroom because we ALL share i
t together. Because my time with them is limited, I want to ensure they get
the most of this time and enjoy it to the best of their abilities.Currently,
we have twenty-two desks of differing sizes, yet the desks are similar to th
e ones the students will use in middle school. We also have a kidney table w
ith crates for seating. I allow my students to choose their own spots while
they are working independently or in groups. More often than not, most of th
em move out of their desks and onto the crates. Believe it or not, this has
proven to be more successful than making them stay at their desks! It is bec
ause of this that I am looking toward the “Flexible Seating” option for my c
lassroom.\r\n The students look forward to their work time so they can move
around the room. I would like to get rid of the constricting desks and move
toward more “fun” seating options. I am requesting various seating so my stu
dents have more options to sit. Currently, I have a stool and a papasan chai
r I inherited from the previous sixth-grade teacher as well as five milk cra
te seats I made, but I would like to give them more options and reduce the c
ompetition for the “good seats”. I am also requesting two rugs as not only m
ore seating options but to make the classroom more welcoming and appealing.
In order for my students to be able to write and complete work without desk
s, I am requesting a class set of clipboards. Finally, due to curriculum tha
t requires groups to work together, I am requesting tables that we can fold
up when we are not using them to leave more room for our flexible seating op
tions.\r\nI know that with more seating options, they will be that much more
excited about coming to school! Thank you for your support in making my clas
sroom one students will remember forever!nannan
==================================================
```

In [17]:

```python
# 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 [18]:

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

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smalle
st students with the biggest enthusiasm for learning. My students learn in m
any different ways using all of our senses and multiple intelligences. I use
a wide range of techniques to help all my students succeed. \r\nStudents in
my class come from a variety of different backgrounds which makes for wonder
ful sharing of experiences and cultures, including Native Americans.\r\nOur
school is a caring community of successful learners which can be seen throug
h collaborative student project based learning in and out of the classroom.
Kindergarteners in my class love to work with hands-on materials and have ma
ny different opportunities to practice a skill before it is mastered. Having
the social skills to work cooperatively with friends is a crucial aspect of
the kindergarten curriculum.Montana is the perfect place to learn about agri
culture and nutrition. My students love to role play in our pretend kitchen
in the early childhood classroom. I have had several kids ask me, \"Can we t
ry cooking with REAL food?\" I will take their idea and create \"Common Core
Cooking Lessons\" where we learn important math and writing concepts while c
ooking delicious healthy food for snack time. My students will have a ground
ed appreciation for the work that went into making the food and knowledge of
where the ingredients came from as well as how it is healthy for their bodie
s. This project would expand our learning of nutrition and agricultural cook
ing recipes by having us peel our own apples to make homemade applesauce, ma
ke our own bread, and mix up healthy plants from our classroom garden in the
spring. We will also create our own cookbooks to be printed and shared with
families. \r\nStudents will gain math and literature skills as well as a lif
e long enjoyment for healthy cooking.nannan
==================================================
```

In [19]:

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

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smallest
students with the biggest enthusiasm for learning. My students learn in many
different ways using all of our senses and multiple intelligences. I use a w
ide range of techniques to help all my students succeed.   Students in my cl
ass come from a variety of different backgrounds which makes for wonderful s
haring of experiences and cultures, including Native Americans.  Our school
is a caring community of successful learners which can be seen through colla
borative student project based learning in and out of the classroom. Kinderg
arteners in my class love to work with hands-on materials and have many diff
erent opportunities to practice a skill before it is mastered. Having the so
cial skills to work cooperatively with friends is a crucial aspect of the ki
ndergarten curriculum.Montana is the perfect place to learn about agricultur
e and nutrition. My students love to role play in our pretend kitchen in the
early childhood classroom. I have had several kids ask me,  Can we try cooki
ng with REAL food?  I will take their idea and create  Common Core Cooking L
essons  where we learn important math and writing concepts while cooking del
icious healthy food for snack time. My students will have a grounded appreci
ation for the work that went into making the food and knowledge of where the
ingredients came from as well as how it is healthy for their bodies. This pr
oject would expand our learning of nutrition and agricultural cooking recipe
s by having us peel our own apples to make homemade applesauce, make our own
bread, and mix up healthy plants from our classroom garden in the spring. We
will also create our own cookbooks to be printed and shared with families.
Students will gain math and literature skills as well as a life long enjoyme
nt for healthy cooking.nannan
```

In [20]:

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

```
 A person is a person no matter how small Dr Seuss I teach the smallest stud
ents with the biggest enthusiasm for learning My students learn in many diff
erent ways using all of our senses and multiple intelligences I use a wide r
ange of techniques to help all my students succeed Students in my class come
from a variety of different backgrounds which makes for wonderful sharing of
experiences and cultures including Native Americans Our school is a caring c
ommunity of successful learners which can be seen through collaborative stud
ent project based learning in and out of the classroom Kindergarteners in my
class love to work with hands on materials and have many different opportuni
ties to practice a skill before it is mastered Having the social skills to w
ork cooperatively with friends is a crucial aspect of the kindergarten curri
culum Montana is the perfect place to learn about agriculture and nutrition
My students love to role play in our pretend kitchen in the early childhood
classroom I have had several kids ask me Can we try cooking with REAL food I
will take their idea and create Common Core Cooking Lessons where we learn i
mportant math and writing concepts while cooking delicious healthy food for
snack time My students will have a grounded appreciation for the work that w
ent into making the food and knowledge of where the ingredients came from as
well as how it is healthy for their bodies This project would expand our lea
rning of nutrition and agricultural cooking recipes by having us peel our ow
n apples to make homemade applesauce make our own bread and mix up healthy p
lants from our classroom garden in the spring We will also create our own co
okbooks to be printed and shared with families Students will gain math and l
iterature skills as well as a life long enjoyment for healthy cooking nannan
```

In [21]:

```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'c
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'dc
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"]
```

In [22]:

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

```
100%|███████████████████████████████████████████████████████████████████████
█| 109248/109248 [01:06<00:00, 1649.86it/s]
```

In [23]:

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

Out[23]:

```
'person person no matter small dr seuss teach smallest students biggest enth
usiasm learning students learn many different ways using senses multiple int
elligences use wide range techniques help students succeed students class co
me variety different backgrounds makes wonderful sharing experiences culture
s including native americans school caring community successful learners see
n collaborative student project based learning classroom kindergarteners cla
ss love work hands materials many different opportunities practice skill mas
tered social skills work cooperatively friends crucial aspect kindergarten c
urriculum montana perfect place learn agriculture nutrition students love ro
le play pretend kitchen early childhood classroom several kids ask try cooki
ng real food take idea create common core cooking lessons learn important ma
th writing concepts cooking delicious healthy food snack time students groun
ded appreciation work went making food knowledge ingredients came well healt
hy bodies project would expand learning nutrition agricultural cooking recip
es us peel apples make homemade applesauce make bread mix healthy plants cla
ssroom garden spring also create cookbooks printed shared families students 
gain math literature skills well life long enjoyment healthy cooking nannan'
```

1.4 Preprocessing of `project_title`

In [24]:

```
# similarly you can preprocess the titles also

preprocessed_title=[]
for tit in tqdm(project_data['project_title'].values):
    tit = decontracted(tit)
    tit = tit.replace('\\r', ' ')
    tit = tit.replace('\\"', ' ')
    tit = tit.replace('\\n', ' ')
    tit = re.sub('[^A-Za-z0-9]+', ' ', tit)
    tit = ' '.join(e for e in tit.split() if e not in stopwords)
    preprocessed_title.append(tit.lower().strip())
print(preprocessed_title[23])
```

```
100%|████████████████████████████████████████████████████████████████
| 109248/109248 [00:03<00:00, 34246.58it/s]

techies training
```

1.5 Preparing data for models

In [25]:

```
project_data.columns
```

Out[25]:

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

we are going to consider

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

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

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

Assignment 3: Apply KNN

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

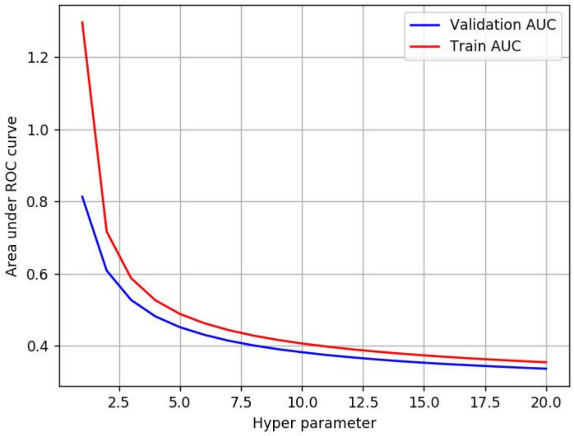
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)
    - Set 4: categorical,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

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

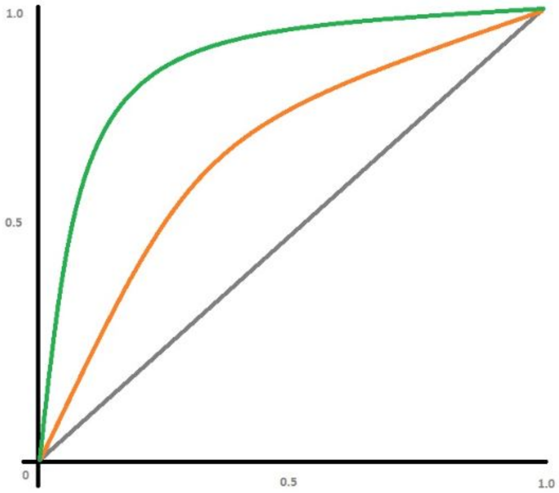
    - Find the best hyper parameter which results in the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using k-fold cross validation (or) simple cross validation data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

3. **Representation of results**

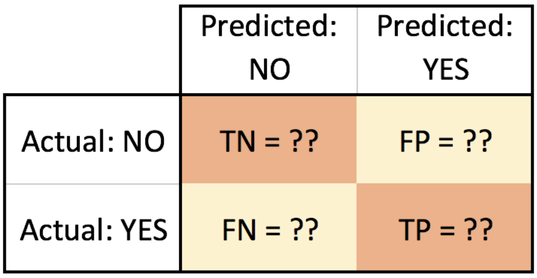
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure



    - Once you find the best hyper parameter, you need to train your model-M using the best hyper-param. Now, find the AUC on test data and plot the ROC curve on both train and test using model-M.



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



	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

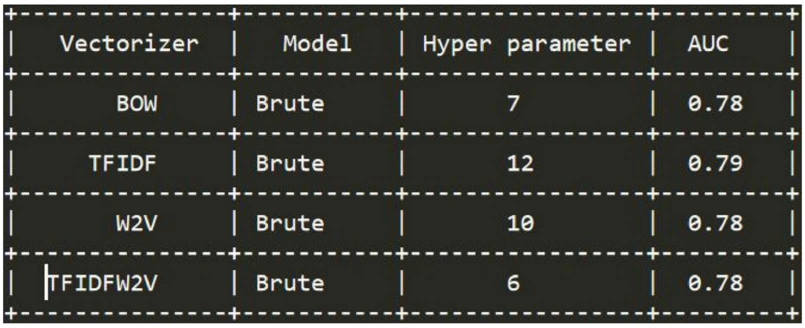
4. **[Task-2]**

- Select top 2000 features from feature Set 2 using `SelectKBest` [(https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html)](https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html) and then apply KNN on top of these features

-
  ```
  from sklearn.datasets import load_digits
  from sklearn.feature_selection import SelectKBest, chi2
  X, y = load_digits(return_X_y=True)
  X.shape
  X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
  X_new.shape
  ========
  output:
  (1797, 64)
  (1797, 20)
  ```
- Repeat the steps 2 and 3 on the data matrix after feature selection

5. **Conclusion**

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



```
+--------------+----------+-------------------+---------+
|  Vectorizer  |  Model   | Hyper parameter  |  AUC    |
+--------------+----------+-------------------+---------+
|     BOW      | Brute    |        7          |  0.78   |
+--------------+----------+-------------------+---------+
|    TFIDF     | Brute    |        12         |  0.79   |
+--------------+----------+-------------------+---------+
|     W2V      | Brute    |        10         |  0.78   |
+--------------+----------+-------------------+---------+
|   TFIDFW2V   | Brute    |        6          |  0.78   |
+--------------+----------+-------------------+---------+
```

Note: Data Leakage

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

2. K Nearest Neighbor

In [26]:

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

```
(109248, 18)
```

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

Here we are taking 50,000 points randomly due to limited memory

In [27]:

```
data = project_data[10000:60000]
data.head()
```

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
92145	55875	p056928	be23f76dbd9d2d11e43b12955482e505	Mrs.	MO	2
22514	89116	p255739	de6a362aa7f79b85cef68ea7234a6790	Mrs.	SC	2
97331	115494	p068053	60733d643ae7a27cf0ba2ceb401576e3	Mrs.	TX	2
104654	87469	p120917	03796981cac39e0de90c56d4de922b50	Mrs.	NC	2
41477	51351	p211590	10507c644cb1ed3dbb17cd467dbf1ac5	Ms.	VT	2

In [28]:

```
y = data['project_is_approved'].values
data.drop(['project_is_approved'], axis=1, inplace=True)
```

In [29]:

```
data.head()
```

Out[29]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
92145	55875	p056928	be23f76dbd9d2d11e43b12955482e505	Mrs.	MO	2
22514	89116	p255739	de6a362aa7f79b85cef68ea7234a6790	Mrs.	SC	2
97331	115494	p068053	60733d643ae7a27cf0ba2ceb401576e3	Mrs.	TX	2
104654	87469	p120917	03796981cac39e0de90c56d4de922b50	Mrs.	NC	2
41477	51351	p211590	10507c644cb1ed3dbb17cd467dbf1ac5	Ms.	VT	2

In [30]:

```
X=data
```

In [31]:

```
# 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=
```

2.2 Make Data Model Ready: encoding numerical, categorical features

fit() : used for generating learning model parameters from training data

transform() : parameters generated from fit() method,applied upon model to generate transformed data set.

fit_transform() : combination of fit() and transform() api on same data set

In [32]:

```
# one hot encoding for "School_state "
vectorizer = CountVectorizer(vocabulary=set(project_data.school_state), lowercase=False, bi
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

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

print("After vectorizations")
print("="*50)
print(X_train_school_state_one_hot.shape, y_train.shape)
print(X_cv_school_state_one_hot.shape, y_cv.shape)
print(X_test_school_state_one_hot.shape, y_test.shape)
print("="*50)
print(vectorizer.get_feature_names())
```

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

In [33]:

```
# one hot encoding for "project_grade_category"

pattern = "(?u)\\b[\\w-]+\\b"
vectorizer = CountVectorizer(token_pattern=pattern, lowercase=False, binary=True)
vectorizer.fit(X_train['project_grade_category'].values) # fit has to happen only on train

# we use the fitted CountVectorizer to convert the text to vector
X_train_project_grade_category_one_hot = vectorizer.transform(X_train['project_grade_catego
X_cv_project_grade_category_one_hot = vectorizer.transform(X_cv['project_grade_category'].v
X_test_project_grade_category_one_hot = vectorizer.transform(X_test['project_grade_category

print("After vectorizations")
print("="*50)
print(X_train_project_grade_category_one_hot.shape, y_train.shape)
print(X_cv_project_grade_category_one_hot.shape, y_cv.shape)
print(X_test_project_grade_category_one_hot.shape, y_test.shape)
print("="*50)
print(vectorizer.get_feature_names())

type(X_train_project_grade_category_one_hot)
df = pd.DataFrame(X_train_project_grade_category_one_hot.toarray())
df.head()
```

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

Out[33]:

	0	1	2	3	4
0	1	0	0	1	0
1	0	1	0	1	0
2	0	0	0	1	1
3	0	0	1	1	0
4	0	1	0	1	0

In [34]:

```
# one hot encoding for "clean_categories"

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, bina
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data

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

print("After vectorizations")
print("="*50)
print(X_train_clean_categories_one_hot.shape, y_train.shape)
print(X_cv_clean_categories_one_hot.shape, y_cv.shape)
print(X_test_clean_categories_one_hot.shape, y_test.shape)
print("="*50)
print(vectorizer.get_feature_names())
```

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

In [35]:

```
# one hot encoding for "clean_subcategories"

vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train dat

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_subcategories_one_hot = vectorizer.transform(X_train['clean_subcategories'].v
X_cv_clean_subcategories_one_hot = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_clean_subcategories_one_hot = vectorizer.transform(X_test['clean_subcategories'].val

print("After vectorizations")
print("="*50)
print(X_train_clean_subcategories_one_hot.shape, y_train.shape)
print(X_cv_clean_subcategories_one_hot.shape, y_cv.shape)
print(X_test_clean_subcategories_one_hot.shape, y_test.shape)
print("="*50)
print(vectorizer.get_feature_names())
```

```
After vectorizations
==================================================
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
==================================================
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducat
ion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Characte
rEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_
Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness',
'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
```

In [36]:

```
# one hot encoding for "teacher_prefix"
import re
Clean_prefix = []

for prefix in (project_data['teacher_prefix'].values):
    prefix = re.sub('[^A-Za-z0-9]+', ' ', str(prefix))
    Clean_prefix.append(prefix)

vectorizer = CountVectorizer(vocabulary=set(Clean_prefix),lowercase=False, binary=True)

vectorizer.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_prefix_one_hot = vectorizer.transform(X_train['teacher_prefix'].values)
X_cv_teacher_prefix_one_hot = vectorizer.transform(X_cv['teacher_prefix'].values.astype("U"
X_test_teacher_prefix_one_hot = vectorizer.transform(X_test['teacher_prefix'].values.astype

print("After vectorizations")
print("="*50)
print(X_train_teacher_prefix_one_hot.shape, y_train.shape)
print(X_cv_teacher_prefix_one_hot.shape, y_cv.shape)
print(X_test_teacher_prefix_one_hot.shape, y_test.shape)
print("="*50)
print(vectorizer.get_feature_names())
```

```
After vectorizations
==================================================
(22445, 6) (22445,)
(11055, 6) (11055,)
(16500, 6) (16500,)
==================================================
['Dr ', 'Mr ', 'Mrs ', 'Ms ', 'Teacher', 'nan']
```

In [41]:

```
# vectorizing numerical features "teacher_number_of_previously_posted_projects"

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

X_train_teacher_number_of_previously_posted_projects = normalizer.transform(X_train['teache
X_cv_teacher_number_of_previously_posted_projects = normalizer.transform(X_cv['teacher_numb
X_test_teacher_number_of_previously_posted_projects = normalizer.transform(X_test['teacher_

print("After vectorizations")
print("="*50)
print(X_train_teacher_number_of_previously_posted_projects.shape, y_train.shape)
print(X_cv_teacher_number_of_previously_posted_projects.shape, y_cv.shape)
print(X_test_teacher_number_of_previously_posted_projects.shape, y_test.shape)
```

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

In [48]:

```
# vectorizing numerical features "price"

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

X_train_price = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_cv_price = normalizer.transform(X_cv['price'].values.reshape(-1,1))
X_test_price = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print("="*50)
print(X_train_price.shape, y_train.shape)
print(X_cv_price.shape, y_cv.shape)
print(X_test_price.shape, y_test.shape)
```

```
(48631, 1) (48631,)
(20842, 1) (20842,)
(29775, 1) (29775,)
```

In [49]:

```
# vectorizing numerical features "quantity"

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

X_train_quantity = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_quantity = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quantity = normalizer.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print("="*50)
print(X_train_quantity.shape, y_train.shape)
print(X_cv_quantity.shape, y_cv.shape)
print(X_test_quantity.shape, y_test.shape)
```

```
(48631, 1) (48631,)
(20842, 1) (20842,)
(29775, 1) (29775,)
```

2.3 Make Data Model Ready: encoding essay, and project_title

Bag of Words

In [35]:

```
# BOW for essay
from sklearn.feature_extraction.text import CountVectorizer
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_bow_essay = vectorizer.transform(X_train['essay'].values)
X_cv_bow_essay = vectorizer.transform(X_cv['essay'].values)
X_test_bow_essay = vectorizer.transform(X_test['essay'].values)

print("After vectorizations")
print("="*50)
print(X_train_bow_essay.shape, y_train.shape)
print(X_cv_bow_essay.shape, y_cv.shape)
print(X_test_bow_essay.shape, y_test.shape)
```

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

In [36]:

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

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

print("After vectorizations")
print("="*50)
print(X_train_bow_title.shape, y_train.shape)
print(X_cv_bow_title.shape, y_cv.shape)
print(X_test_bow_title.shape, y_test.shape)
```

```
After vectorizations
==================================================
(22445, 2708) (22445,)
(11055, 2708) (11055,)
(16500, 2708) (16500,)
```

TF-IDF

In [37]:

```
#TF-idf for "essay"

from sklearn.feature_extraction.text import TfidfVectorizer
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_tfidf_essay = vectorizer.transform(X_train['essay'].values)
X_cv_tfidf_essay = vectorizer.transform(X_cv['essay'].values)
X_test_tfidf_essay = vectorizer.transform(X_test['essay'].values)

print("After vectorizations")
print("="*50)
print(X_train_tfidf_essay.shape, y_train.shape)
print(X_cv_tfidf_essay.shape, y_cv.shape)
print(X_test_tfidf_essay.shape, y_test.shape)
```

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

In [38]:

```
#TF-idf for "Project_title"

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

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

print("After vectorizations")
print("="*50)
print(X_train_tfidf_title.shape, y_train.shape)
print(X_cv_tfidf_title.shape, y_cv.shape)
print(X_test_tfidf_title.shape, y_test.shape)
```

```
After vectorizations
==================================================
(22445, 2708) (22445,)
(11055, 2708) (11055,)
(16500, 2708) (16500,)
```

Avg-W2V

In [39]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickl
# 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 [40]:

```
# average Word2Vec for "essay" in training data

X_train_avgw2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # 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
    X_train_avgw2v_essay.append(vector)

print(len(X_train_avgw2v_essay))
print(len(X_train_avgw2v_essay[1]))
```

```
100%|███████████████████████████████████
███| 22445/22445 [00:09<00:00, 2272.48it/s]

22445
300
```

In [41]:

```
# average Word2Vec for "essay" in crossvalidation data

X_cv_avgw2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay']): # 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
    X_cv_avgw2v_essay.append(vector)

print(len(X_cv_avgw2v_essay))
print(len(X_cv_avgw2v_essay[1]))
```

```
100%|███████████████████████████████████
███| 11055/11055 [00:04<00:00, 2279.38it/s]

11055
300
```

In [42]:

```
# average Word2Vec for "essay" in test data

X_test_avgw2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay']): # 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
    X_test_avgw2v_essay.append(vector)

print(len(X_test_avgw2v_essay))
print(len(X_test_avgw2v_essay[1]))
```

```
100%|████████████████████████████████████████████████████████████████████
███| 16500/16500 [00:07<00:00, 2307.50it/s]

16500
300
```

In [43]:

```
# average Word2Vec for "project_title" in training data

X_train_avgw2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['project_title']): # 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
    X_train_avgw2v_title.append(vector)

print(len(X_train_avgw2v_title))
print(len(X_train_avgw2v_title[1]))
```

```
100%|████████████████████████████████████████████████████████████████████
██| 22445/22445 [00:00<00:00, 72918.17it/s]

22445
300
```

In [44]:

```
# average Word2Vec for "project_title" in crossvalidation data

X_cv_avgw2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['project_title']): # 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
    X_cv_avgw2v_title.append(vector)

print(len(X_cv_avgw2v_title))
print(len(X_cv_avgw2v_title[1]))
```

```
100%|█████████████████████████████████████████████████████████████████
█| 11055/11055 [00:00<00:00, 81938.67it/s]

11055
300
```

In [45]:

```
# average Word2Vec for "project_title" in test data

X_test_avgw2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['project_title']): # 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
    X_test_avgw2v_title.append(vector)

print(len(X_test_avgw2v_title))
print(len(X_test_avgw2v_title[1]))
```

```
100%|█████████████████████████████████████████████████████████████████
█| 16500/16500 [00:00<00:00, 83383.86it/s]

16500
300
```

TF-IDF Weighted W2v

In [46]:

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

In [47]:

```
# TF-IDF weighted Word2Vec for "essay" in training data

X_train_weightw2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # 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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_weightw2v_essay.append(vector)

print(len(X_train_weightw2v_essay))
print(len(X_train_weightw2v_essay[0]))
```

```
100%|████████████████████████████████████
██| 22445/22445 [01:00<00:00, 373.82it/s]

22445
300
```

In [48]:

```
# TF-IDF weighted Word2Vec for "essay" in cross validation data

X_cv_weightw2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay']): # 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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_cv_weightw2v_essay.append(vector)

print(len(X_cv_weightw2v_essay))
print(len(X_cv_weightw2v_essay[0]))
```

```
100%|████████████████████████████████████
██| 11055/11055 [00:32<00:00, 339.67it/s]

11055
300
```

In [49]:

```
# TF-IDF weighted Word2Vec for "essay" in cross test data

X_test_weightw2v_essay = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay']): # 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((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_weightw2v_essay.append(vector)

print(len(X_test_weightw2v_essay))
print(len(X_test_weightw2v_essay[0]))
```

```
100%|████████████████████████████████████████████████████████████
███| 16500/16500 [00:46<00:00, 352.82it/s]

16500
300
```

In [ ]:

```

```

In [50]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_title)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary_title = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [51]:

```
# TF-IDF weighted Word2Vec for "project_title" in training data

X_train_weightw2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['project_title']): # 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((sentenc
            tf_idf = dictionary_title[word]*(sentence.count(word)/len(sentence.split())) #
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_train_weightw2v_title.append(vector)

print(len(X_train_weightw2v_title))
print(len(X_train_weightw2v_title[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██| 22445/22445 [00:00<00:00, 53219.96it/s]

22445
300
```

In [52]:

```
# TF-IDF weighted Word2Vec for "project_title" in cross validation data

X_cv_weightw2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['project_title']): # 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((sentenc
            tf_idf = dictionary_title[word]*(sentence.count(word)/len(sentence.split())) #
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_cv_weightw2v_title.append(vector)

print(len(X_cv_weightw2v_title))
print(len(X_cv_weightw2v_title[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██| 11055/11055 [00:00<00:00, 52178.12it/s]

11055
300
```

In [53]:

```
# TF-IDF weighted Word2Vec for "project_title" in training data

X_test_weightw2v_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['project_title']): # 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((sentenc
            tf_idf = dictionary_title[word]*(sentence.count(word)/len(sentence.split())) #
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_test_weightw2v_title.append(vector)

print(len(X_test_weightw2v_title))
print(len(X_test_weightw2v_title[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
█| 16500/16500 [00:00<00:00, 58964.58it/s]

16500
300
```

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

2.4.1 Applying KNN brute force on BOW, SET 1

In [65]:

```
# concatinating all the features
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

from scipy.sparse import hstack
X_bow_train = hstack((X_train_bow_essay, X_train_bow_title, X_train_school_state_one_hot, X
X_bow_cv = hstack((X_cv_bow_essay, X_cv_bow_title, X_cv_school_state_one_hot, X_cv_project_
X_bow_test = hstack((X_test_bow_essay, X_test_bow_title, X_test_school_state_one_hot, X_tes

print("Final Data matrix")
print("="*50)
print(X_bow_train.shape, y_train.shape)
print(X_bow_cv.shape, y_cv.shape)
print(X_bow_test.shape, y_test.shape)
```

```
Final Data matrix
==================================================
(22445, 7797) (22445,)
(11055, 7797) (11055,)
(16500, 7797) (16500,)
```

Hyper parameter tuning using simple for loop

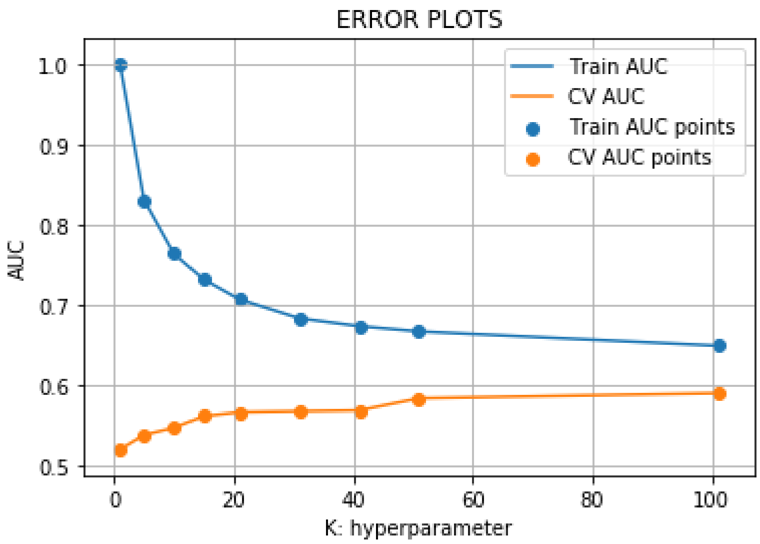
In [65]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 4900
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [68]:

```
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
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.

"""


bow_train_auc = []
bow_cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_bow_train, y_train)

    y_train_bow_pred = batch_predict(neigh, X_bow_train)
    y_cv_bow_pred = batch_predict(neigh, X_bow_cv)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    bow_train_auc.append(roc_auc_score(y_train,y_train_bow_pred))
    bow_cv_auc.append(roc_auc_score(y_cv, y_cv_bow_pred))

plt.plot(K, bow_train_auc, label='Train AUC')
plt.plot(K, bow_cv_auc, label='CV AUC')

plt.scatter(K, bow_train_auc, label='Train AUC points')
plt.scatter(K, bow_cv_auc, label='CV AUC points')

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

```
100%|███████████████████████████████████
██████████████████████████████| 9/9 [22:43<00:
00, 152.05s/it]
```



From the Above Plot we can say that the Best K will be Around 50 because after that the Graph of CV_AUC is almost constant after 51 hence, LET USE CONSIDER OUR BEST K = 51

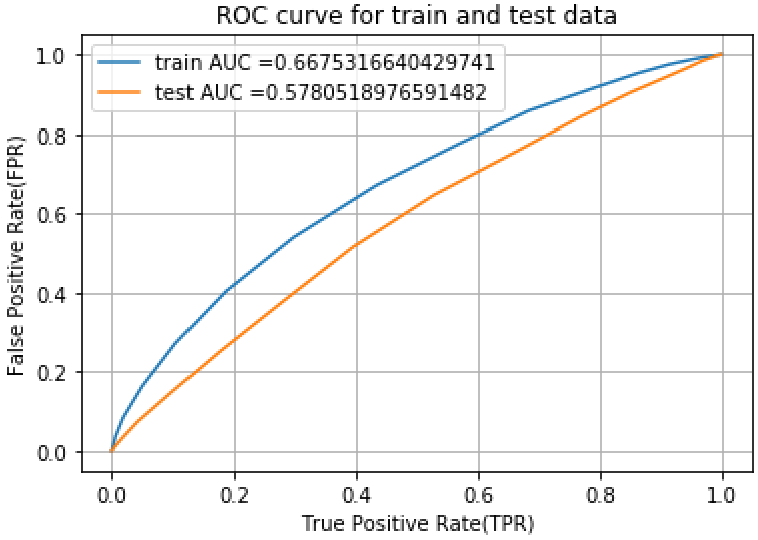
In [73]:

```
best_k_bow = 51
```

Roc Curve

In [74]:

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

neigh = KNeighborsClassifier(n_neighbors=best_k_bow)
neigh.fit(X_bow_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_bow_pred = batch_predict(neigh, X_bow_train)
y_test_bow_pred = batch_predict(neigh, X_bow_test)

train_bow_fpr, train_bow_tpr, train_bow_thresholds = roc_curve(y_train, y_train_bow_pred)
test_bow_fpr, test_bow_tpr, test_bow_thresholds= roc_curve(y_test, y_test_bow_pred)

plt.plot(train_bow_fpr, train_bow_tpr, label="train AUC ="+str(auc(train_bow_fpr, train_bow

bow_test_auc = auc(test_bow_fpr, test_bow_tpr)
plt.plot(test_bow_fpr, test_bow_tpr, label="test AUC ="+str(bow_test_auc))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



From the above curve it is clearly visible that the Test data AUC is below than the Train data AUC i.e. model is not performing well and their is around 10% difference between them.

Confusion Matrix

In [92]:

```
def predict(proba, threshould, fpr, tpr):

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

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

Train Data

In [93]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_bow_pred, train_bow_thresholds, train_bow_f
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24557616955289074 for threshold 0.824
[[ 2074  1587]
 [ 6167 12617]]
```

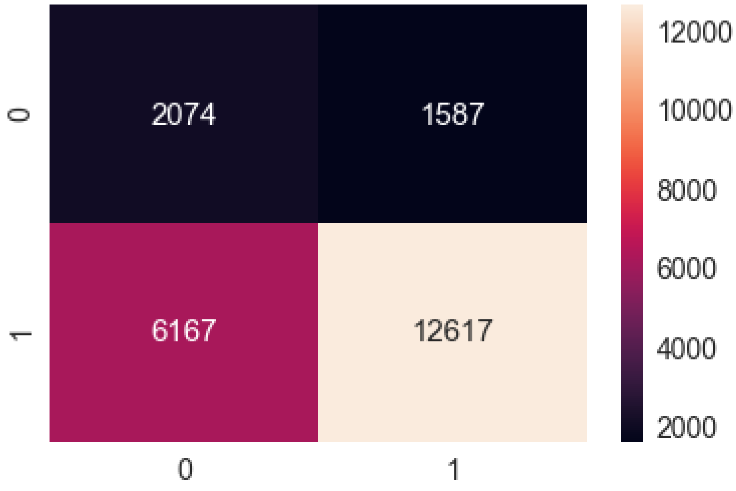
In [96]:

```
cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_bow_pred, train_bow_thres
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_train, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24557616955289074 for threshold 0.824
```

Out[96]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x188ab278ac8>
```



Test Data

In [97]:

```
print("test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_bow_pred, test_bow_thresholds, test_bow_fpr,
```

```
test confusion matrix
the maximum value of tpr*(1-fpr) 0.2492128336731119 for threshold 0.824
[[1270 1421]
 [4873 8936]]
```

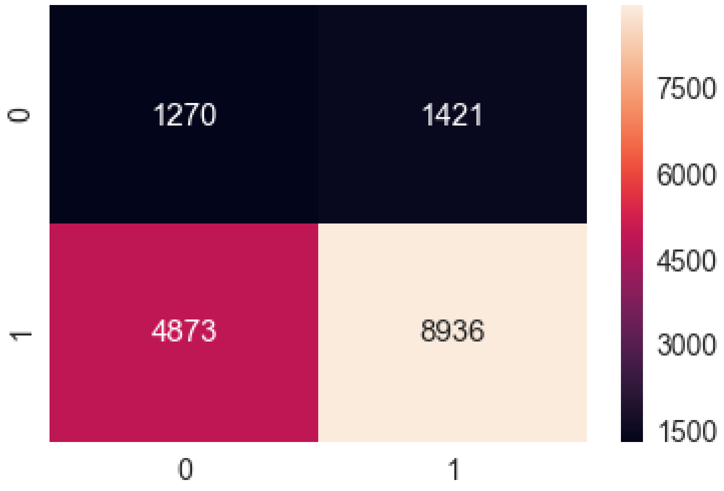
In [100]:

```
cm_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_bow_pred, test_bow_thresholc
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_test, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2492128336731119 for threshold 0.824
```

Out[100]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18886245eb8>
```



2.4.2 Applying KNN brute force on TFIDF, SET 2

In [63]:

```
# concatinating all the features
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

from scipy.sparse import hstack
X_tfidf_train = hstack((X_train_tfidf_essay, X_train_tfidf_title, X_train_school_state_one_
X_tfidf_cv = hstack((X_cv_tfidf_essay, X_cv_tfidf_title, X_cv_school_state_one_hot, X_cv_pr
X_tfidf_test = hstack((X_test_tfidf_essay, X_test_tfidf_title, X_test_school_state_one_hot,

print("Final Data matrix")
print("="*50)
print(X_tfidf_train.shape, y_train.shape)
print(X_tfidf_cv.shape, y_cv.shape)
print(X_tfidf_test.shape, y_test.shape)
```

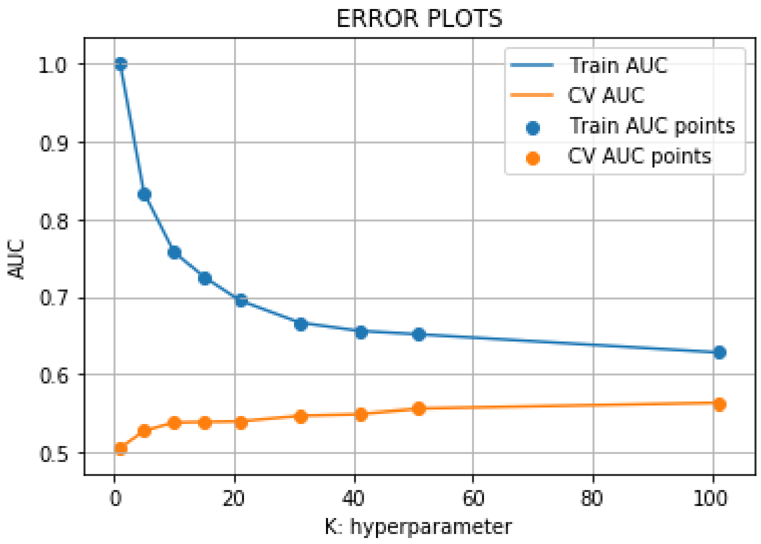
```
Final Data matrix
==================================================
(22445, 7810) (22445,)
(11055, 7810) (11055,)
(16500, 7810) (16500,)
```

Hyper parameter tuning using simple for loop

In [66]:

```
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
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.

"""


tfidf_train_auc = []
tfidf_cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tfidf_train, y_train)

    y_train_tfidf_pred = batch_predict(neigh, X_tfidf_train)
    y_cv_tfidf_pred = batch_predict(neigh, X_tfidf_cv)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    tfidf_train_auc.append(roc_auc_score(y_train,y_train_tfidf_pred))
    tfidf_cv_auc.append(roc_auc_score(y_cv, y_cv_tfidf_pred))

plt.plot(K, tfidf_train_auc, label='Train AUC')
plt.plot(K, tfidf_cv_auc, label='CV AUC')

plt.scatter(K, tfidf_train_auc, label='Train AUC points')
plt.scatter(K, tfidf_cv_auc, label='CV AUC points')

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

```
100%|███████████████████████████████████████████████
███████| 9/9 [23:14<00:00, 157.18s/it]
```



From the above plot, it is again observed that our K is around 50 as after that the CV_AUC curve is nearly constant. So, we can take our K to be 51.

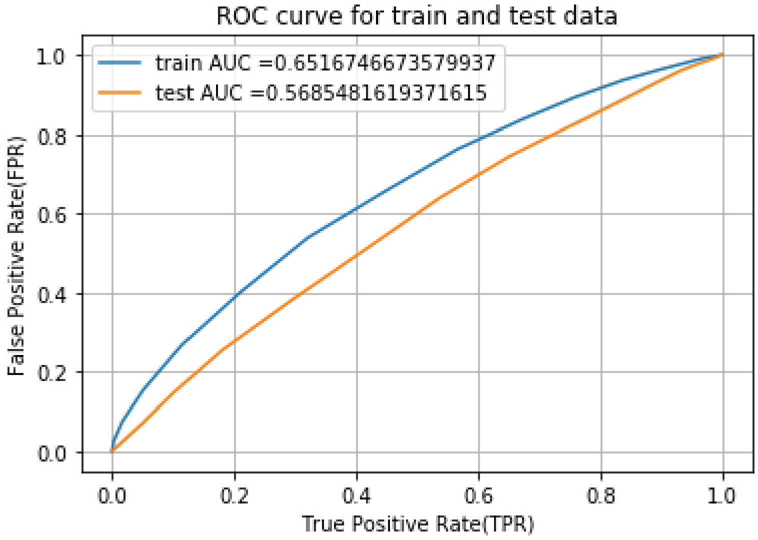
In [67]:

```
best_k_tfidf = 51
```

ROC curve

In [68]:

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

neigh = KNeighborsClassifier(n_neighbors=best_k_tfidf)
neigh.fit(X_tfidf_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_tfidf_pred = batch_predict(neigh, X_tfidf_train)
y_test_tfidf_pred = batch_predict(neigh, X_tfidf_test)

train_tfidf_fpr, train_tfidf_tpr, train_tfidf_thresholds = roc_curve(y_train, y_train_tfidf
test_tfidf_fpr, test_tfidf_tpr, test_tfidf_thresholds= roc_curve(y_test, y_test_tfidf_pred)

plt.plot(train_tfidf_fpr, train_tfidf_tpr, label="train AUC ="+str(auc(train_tfidf_fpr, tra

tfidf_test_auc = auc(test_tfidf_fpr, test_tfidf_tpr)
plt.plot(test_tfidf_fpr, test_tfidf_tpr, label="test AUC ="+str(tfidf_test_auc))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



It is clearly see that AUC for train data is high as compared to the test data. So, it is not performing well & also giving results same as Bag of Words and there is around 10 percent difference between train and test Auc.

Confusion matrix

Train Data

In [101]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_tfidf_pred, train_tfidf_thresholds, train_t
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24759602776141112 for threshold 0.824
[[ 1651  2010]
 [ 4612 14172]]
```

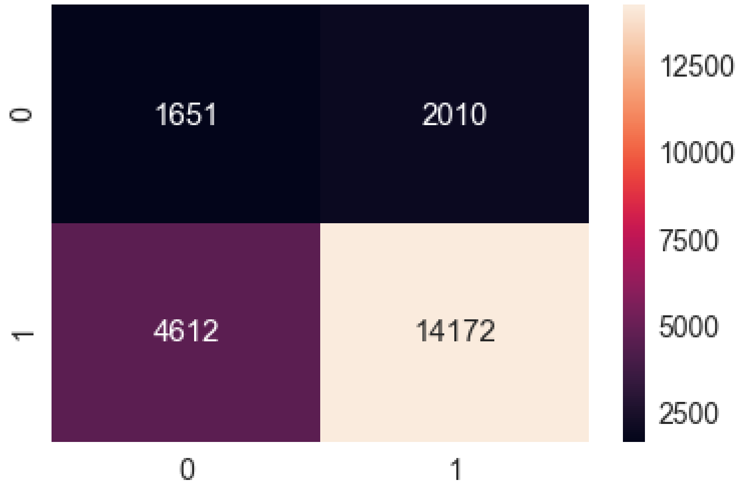
In [102]:

```
cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_tfidf_pred, train_tfidf_t
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_train, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24759602776141112 for threshold 0.824
```

Out[102]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x188862bcc50>
```



Test Data

In [ ]:

```
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_tfidf_pred, train_tfidf_thresholds, train_t
```

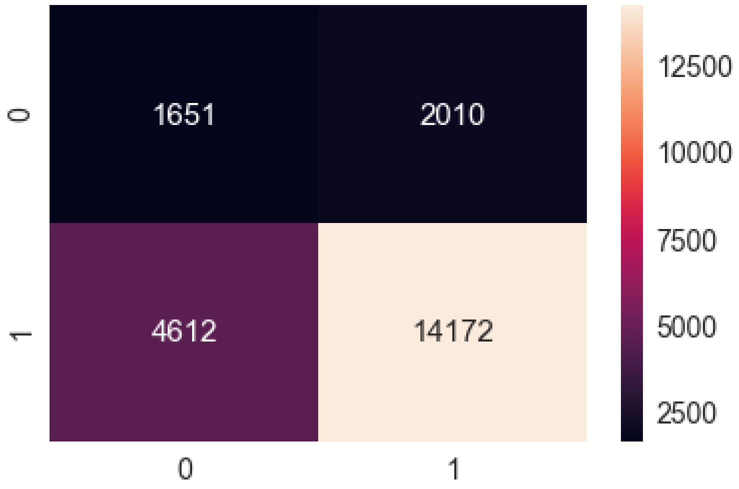
In [103]:

```
cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_tfidf_pred, train_tfidf_t
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_train, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24759602776141112 for threshold 0.824
```

Out[103]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18886335d68>
```



2.4.3 Applying KNN brute force on AVG W2V, SET 3

In [104]:

```
# concatinating all the features
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

from scipy.sparse import hstack
X_avgw2v_train = hstack((X_train_avgw2v_essay, X_train_avgw2v_title, X_train_school_state_c
X_avgw2v_cv = hstack((X_cv_avgw2v_essay, X_cv_avgw2v_title, X_cv_school_state_one_hot, X_cv
X_avgw2v_test = hstack((X_test_avgw2v_essay, X_test_avgw2v_title, X_test_school_state_one_h

print("Final Data matrix")
print("="*50)
print(X_avgw2v_train.shape, y_train.shape)
print(X_avgw2v_cv.shape, y_cv.shape)
print(X_avgw2v_test.shape, y_test.shape)
```

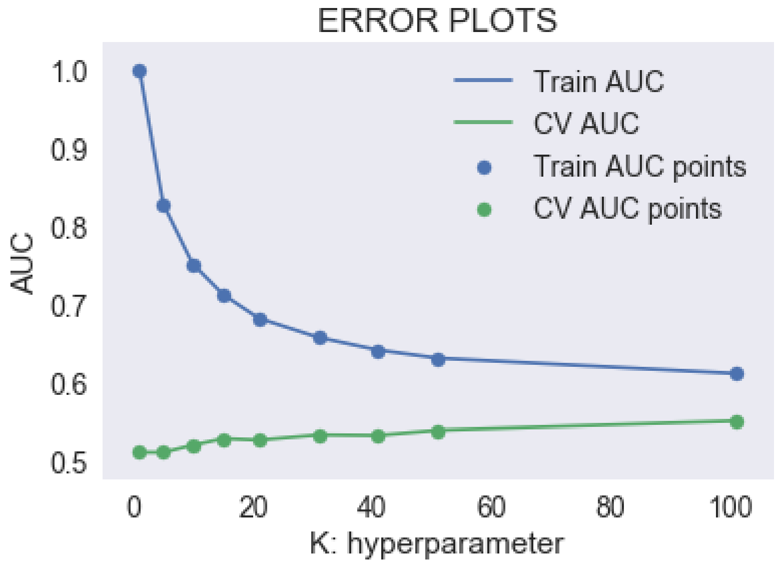
```
Final Data matrix
==================================================
(22445, 701) (22445,)
(11055, 701) (11055,)
(16500, 701) (16500,)
```

Hyper parameter Tuning using simple for loop

In [105]:

```
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
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.

"""


avgw2v_train_auc = []
avgw2v_cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_avgw2v_train, y_train)

    y_train_avgw2v_pred = batch_predict(neigh, X_avgw2v_train)
    y_cv_avgw2v_pred = batch_predict(neigh, X_avgw2v_cv)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    avgw2v_train_auc.append(roc_auc_score(y_train,y_train_avgw2v_pred))
    avgw2v_cv_auc.append(roc_auc_score(y_cv, y_cv_avgw2v_pred))

plt.plot(K, avgw2v_train_auc, label='Train AUC')
plt.plot(K, avgw2v_cv_auc, label='CV AUC')

plt.scatter(K, avgw2v_train_auc, label='Train AUC points')
plt.scatter(K, avgw2v_cv_auc, label='CV AUC points')

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

```
100%|██████████████████████████████████████████████████
█████████████████████████████████████████| 9/9 [1:44:31<00:
00, 696.62s/it]
```



From the above Error plots it is observed that our hyperparameter k is equal to 51 as after that the Error rates are decreasing Very Slowly.

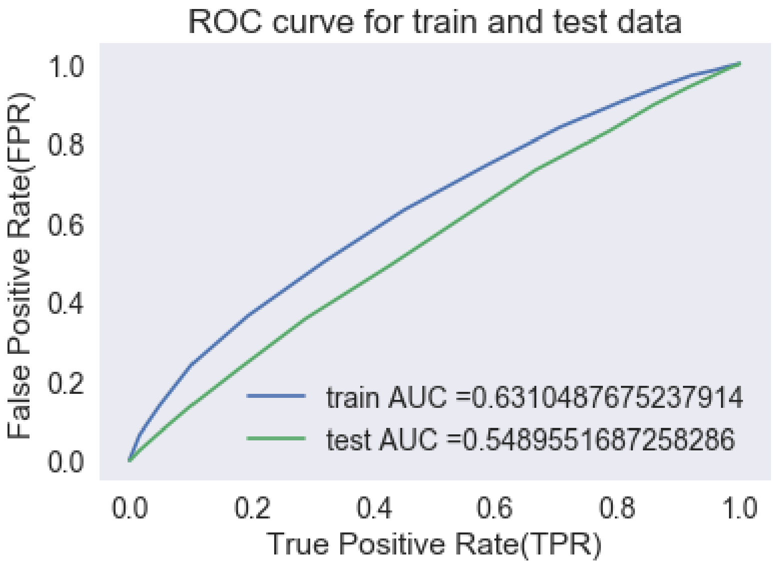
In [107]:

```
best_k_avgw2v = 51
```

ROC Curve

In [108]:

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

neigh = KNeighborsClassifier(n_neighbors=best_k_avgw2v)
neigh.fit(X_avgw2v_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_avgw2v_pred = batch_predict(neigh, X_avgw2v_train)
y_test_avgw2v_pred = batch_predict(neigh, X_avgw2v_test)

train_avgw2v_fpr, train_avgw2v_tpr, train_avgw2v_thresholds = roc_curve(y_train, y_train_av
test_avgw2v_fpr, test_avgw2v_tpr, test_avgw2v_thresholds= roc_curve(y_test, y_test_avgw2v_p

plt.plot(train_avgw2v_fpr, train_avgw2v_tpr, label="train AUC ="+str(auc(train_avgw2v_fpr,

avgw2v_test_auc = auc(test_avgw2v_fpr, test_avgw2v_tpr)
plt.plot(test_avgw2v_fpr, test_avgw2v_tpr, label="test AUC ="+str(avgw2v_test_auc))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



It is clearly see that AUC for train data is high as compared to the test data. So, it is not performing well & also giving results same as Bag of Words & TFIDF and there is around 10 percent difference between train and test Auc.

Confusion matrix

Train Data

In [109]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_avgw2v_pred, train_avgw2v_thresholds, trai
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2475421589070024 for threshold 0.843
[[ 2012  1649]
 [ 6923 11861]]
```

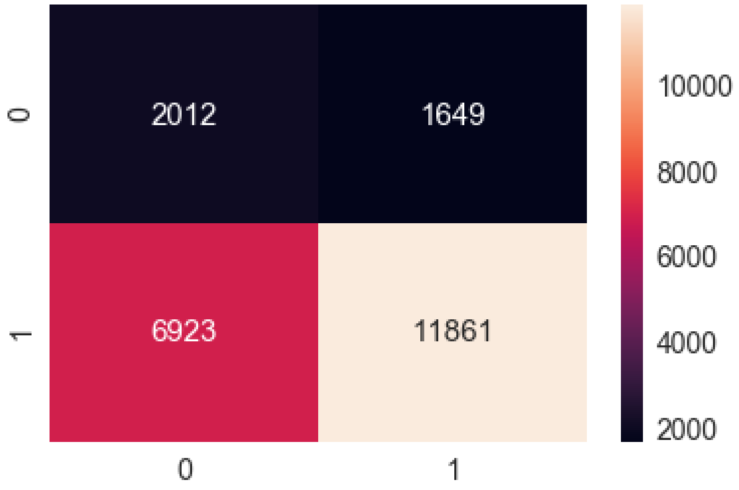
In [110]:

```
cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_avgw2v_pred, train_avgw2v
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_train, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2475421589070024 for threshold 0.843
```

Out[110]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x18887ffdd68>
```



Test Data

In [113]:

```
print("test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_avgw2v_pred, test_avgw2v_thresholds, test_avg
```

```
test confusion matrix
the maximum value of tpr*(1-fpr) 0.24703620709631083 for threshold 0.843
[[1199 1492]
 [5236 8573]]
```

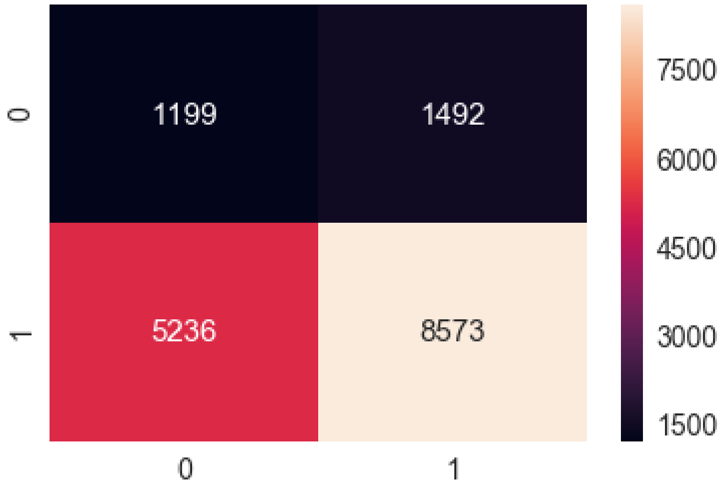
In [114]:

```
cm_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_avgw2v_pred, test_avgw2v_thr
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_test, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24703620709631083 for threshold 0.843
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x188863c5ac8>
```



2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [117]:

```
# concatinating all the features
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

from scipy.sparse import hstack
X_weightw2v_train = hstack((X_train_weightw2v_essay, X_train_weightw2v_title, X_train_schoo
X_weightw2v_cv = hstack((X_cv_weightw2v_essay, X_cv_weightw2v_title, X_cv_school_state_one_
X_weightw2v_test = hstack((X_test_weightw2v_essay, X_test_weightw2v_title, X_test_school_st

print("Final Data matrix")
print("="*50)
print(X_weightw2v_train.shape, y_train.shape)
print(X_weightw2v_cv.shape, y_cv.shape)
print(X_weightw2v_test.shape, y_test.shape)
```

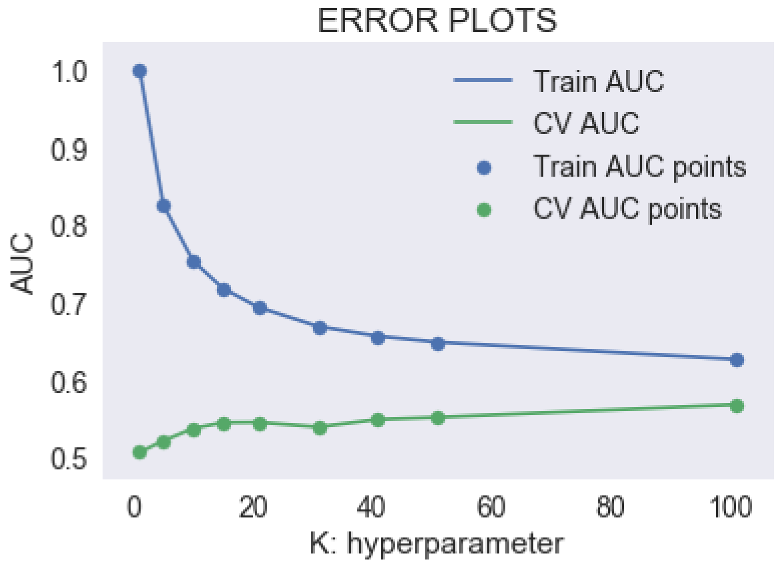
```
Final Data matrix
==================================================
(22445, 701) (22445,)
(11055, 701) (11055,)
(16500, 701) (16500,)
```

Hyper parameter Tuning using simple for loop

In [118]:

```
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
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.

"""


weightw2v_train_auc = []
weightw2v_cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_weightw2v_train, y_train)

    y_train_weightw2v_pred = batch_predict(neigh, X_weightw2v_train)
    y_cv_weightw2v_pred = batch_predict(neigh, X_weightw2v_cv)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    weightw2v_train_auc.append(roc_auc_score(y_train,y_train_weightw2v_pred))
    weightw2v_cv_auc.append(roc_auc_score(y_cv, y_cv_weightw2v_pred))

plt.plot(K, weightw2v_train_auc, label='Train AUC')
plt.plot(K, weightw2v_cv_auc, label='CV AUC')

plt.scatter(K, weightw2v_train_auc, label='Train AUC points')
plt.scatter(K, weightw2v_cv_auc, label='CV AUC points')

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

```
100%|███████████████████████████████████
█████████████████████████████| 9/9 [1:44:03<00:
00, 700.99s/it]
```



Again Error rates are decreasing very slowly after 51 therefore From the Above Error plots we can take our Hyperparameter k as 51.

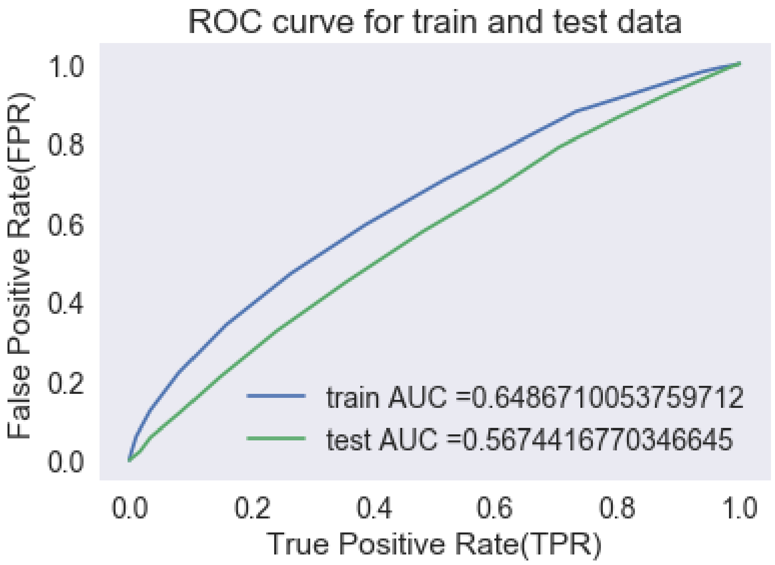
In [141]:

```
best_k_weightw2v = 51
```

ROC curve

In [120]:

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

neigh = KNeighborsClassifier(n_neighbors=best_k_weightw2v)
neigh.fit(X_weightw2v_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_weightw2v_pred = batch_predict(neigh, X_weightw2v_train)
y_test_weightw2v_pred = batch_predict(neigh, X_weightw2v_test)

train_weightw2v_fpr, train_weightw2v_tpr, train_weightw2v_thresholds = roc_curve(y_train, y
test_weightw2v_fpr, test_weightw2v_tpr, test_weightw2v_thresholds= roc_curve(y_test, y_test

plt.plot(train_weightw2v_fpr, train_weightw2v_tpr, label="train AUC ="+str(auc(train_weight

weightw2v_test_auc = auc(test_weightw2v_fpr, test_weightw2v_tpr)
plt.plot(test_weightw2v_fpr, test_weightw2v_tpr, label="test AUC ="+str(weightw2v_test_auc)
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



It is clearly see that AUC for train data is high as compared to the test data. So, it is not performing well & also giving results same as Bag of Words,TFIDF & avgW2V and there is around 10 percent difference between train and test Auc.

Confusion matrix

Train Data

In [121]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_weightw2v_pred, train_weightw2v_thresholds,
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24971780405181826 for threshold 0.824
[[ 1769  1892]
 [ 5481 13303]]
```

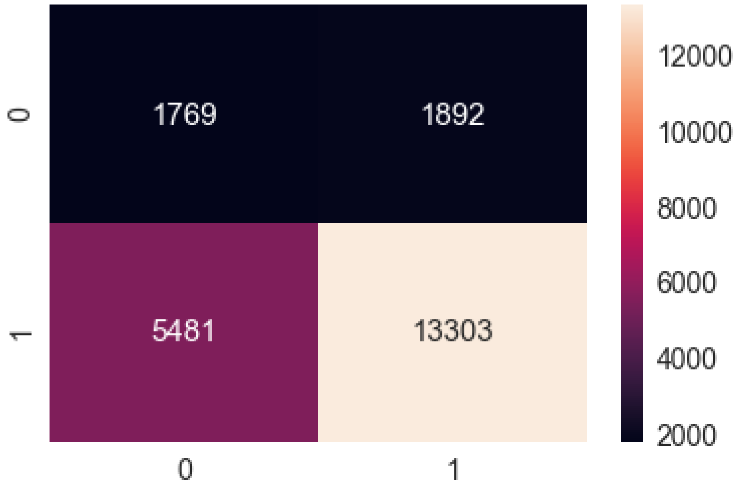
In [122]:

```
cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_weightw2v_pred, train_wei
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_train, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24971780405181826 for threshold 0.824
```

Out[122]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x188ab28af98>
```



Test Data

In [123]:

```
print("test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_weightw2v_pred, test_weightw2v_thresholds, te
```

```
test confusion matrix
the maximum value of tpr*(1-fpr) 0.2496193803449874 for threshold 0.843
[[1398 1293]
 [5847 7962]]
```

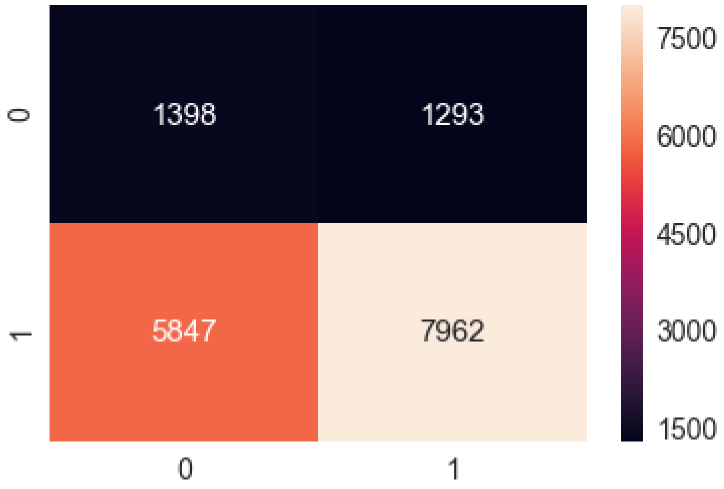
In [124]:

```
cm_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_weightw2v_pred, test_weightw
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_test, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2496193803449874 for threshold 0.843
```

Out[124]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x188b19a4fd0>
```



2.5 Feature selection with `SelectKBest`

In [74]:

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
SKB = SelectKBest(chi2, k=2000).fit(X_tfidf_train, y_train)
X_2000_tfidf_train = SKB.transform(X_tfidf_train)
X_2000_tfidf_cv = SKB.transform(X_tfidf_cv)
X_2000_tfidf_test = SKB.transform(X_tfidf_test)

print(X_2000_tfidf_train.shape)
print(X_2000_tfidf_cv.shape)
print(X_2000_tfidf_test.shape)
```

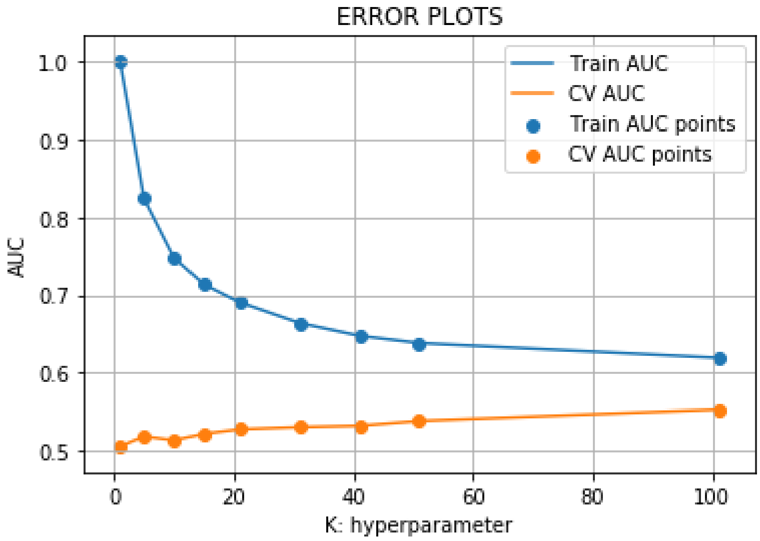
```
(22445, 2000)
(11055, 2000)
(16500, 2000)
```

Hyper parameter Tuning using simple for loop

In [75]:

```
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
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.

"""


tfidf_2000_train_auc = []
tfidf_2000_cv_auc = []
K = [1, 5, 10, 15, 21, 31, 41, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_2000_tfidf_train, y_train)

    y_train_2000_tfidf_pred = batch_predict(neigh, X_2000_tfidf_train)
    y_cv_2000_tfidf_pred = batch_predict(neigh, X_2000_tfidf_cv)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    tfidf_2000_train_auc.append(roc_auc_score(y_train,y_train_2000_tfidf_pred))
    tfidf_2000_cv_auc.append(roc_auc_score(y_cv, y_cv_2000_tfidf_pred))

plt.plot(K, tfidf_2000_train_auc, label='Train AUC')
plt.plot(K, tfidf_2000_cv_auc, label='CV AUC')

plt.scatter(K, tfidf_2000_train_auc, label='Train AUC points')
plt.scatter(K, tfidf_2000_cv_auc, label='CV AUC points')

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

```
100%|██████████████████████████████████████████████████
█████████| 9/9 [07:09<00:00, 47.90s/it]
```



From the above Error plots it is observed that our hyperparameter k is equal to 51 as after that our CV_AUC curve is nearly constant.

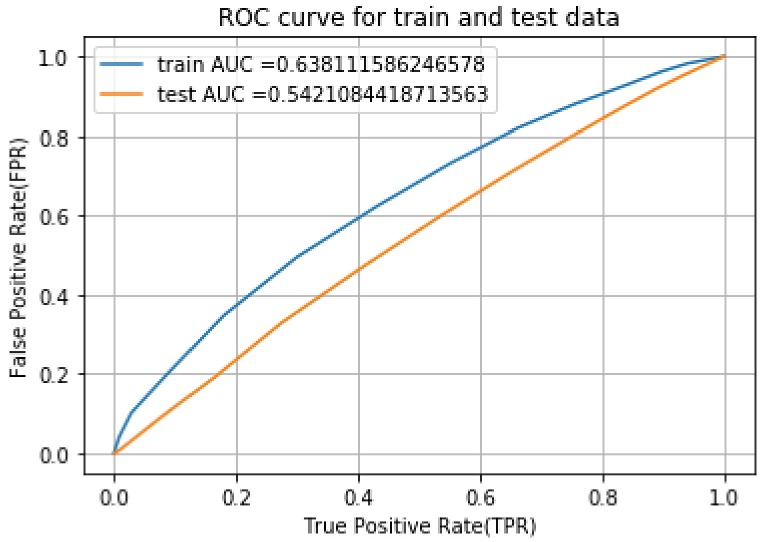
In [76]:

```
best_k_tfidf_2000 =  51
```

Roc Curve

In [77]:

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

neigh = KNeighborsClassifier(n_neighbors=best_k_tfidf_2000)
neigh.fit(X_2000_tfidf_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_2000_tfidf_pred = batch_predict(neigh, X_2000_tfidf_train)
y_test_2000_tfidf_pred = batch_predict(neigh, X_2000_tfidf_test)

train_2000_tfidf_fpr, train_2000_tfidf_tpr, train_2000_tfidf_thresholds = roc_curve(y_trair
test_2000_tfidf_fpr, test_2000_tfidf_tpr, test_2000_tfidf_thresholds= roc_curve(y_test, y_t

plt.plot(train_2000_tfidf_fpr, train_2000_tfidf_tpr, label="train AUC ="+str(auc(train_2000

tfidf_2000_test_auc = auc(test_2000_tfidf_fpr, test_2000_tfidf_tpr)
plt.plot(test_2000_tfidf_fpr, test_2000_tfidf_tpr, label="test AUC ="+str(tfidf_2000_test_a
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



From the Above curves it is Clearly seen that Test Auc is decreased slightly with 2000 features as compared to tfidf with 50,000 Points.

Confusion matrix

Train Data

In [132]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_2000_tfidf_pred, train_2000_tfidf_threshold
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24997446452157704 for threshold 0.824
[[ 1812  1849]
 [ 5674 13110]]
```

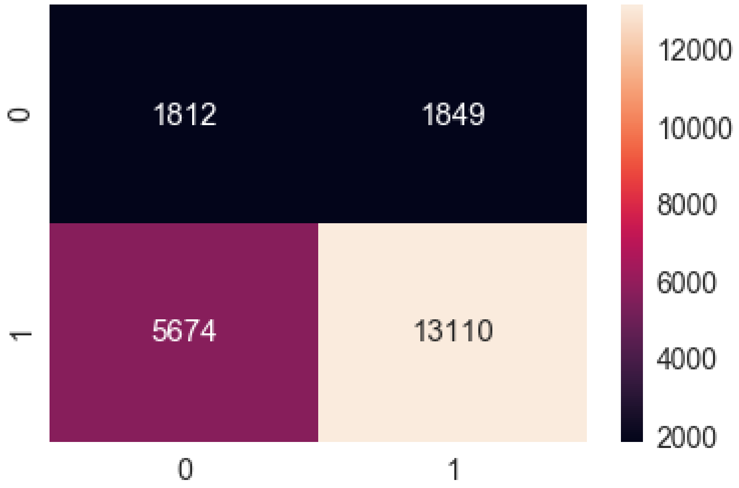
In [133]:

```
cm_train = pd.DataFrame(confusion_matrix(y_train, predict(y_train_2000_tfidf_pred, train_20
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_train, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24997446452157704 for threshold 0.824
```

Out[133]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x188863f6c88>
```



Test Data

In [134]:

```
print("test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_2000_tfidf_pred, test_2000_tfidf_thresholds,
```

```
test confusion matrix
the maximum value of tpr*(1-fpr) 0.2492539854761754 for threshold 0.843
[[1272 1419]
 [6348 7461]]
```

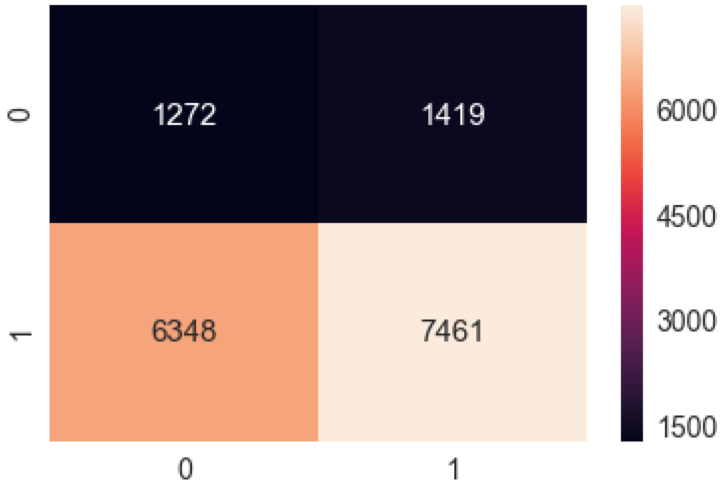
In [135]:

```
cm_test = pd.DataFrame(confusion_matrix(y_test, predict(y_test_2000_tfidf_pred, test_2000_t
sns.set(font_scale=1.4)#for label size
sns.heatmap(cm_test, annot=True, annot_kws={"size": 15}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2492539854761754 for threshold 0.843
```

Out[135]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x188866979e8>
```



3. Conclusion

In [139]:

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

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]

x.add_row(["BOW", "Brute", str(best_k_bow), str(bow_test_auc)])
x.add_row(["TFIDF", "Brute", str(best_k_tfidf), str(tfidf_test_auc)])
x.add_row(["W2V", "Brute", str(best_k_avgw2v), str(avgw2v_test_auc)])
x.add_row(["TFIDFW2V","Brute", str(best_k_weightw2v), str(weightw2v_test_auc)])
x.add_row(["TFIDF_2000","Brute", str(best_k_tfidf_2000), str(tfidf_2000_test_auc)])

print(x)
```

```
+------------+-------+-----------------+--------------------+
| Vectorizer | Model | Hyper Parameter |        AUC         |
+------------+-------+-----------------+--------------------+
|    BOW     | Brute |        51       | 0.5780518976591482 |
|   TFIDF    | Brute |        51       | 0.5637583904356992 |
|    W2V     | Brute |        51       | 0.5489551687258286 |
|  TFIDFW2V  | Brute |        51       | 0.5674416770346645 |
| TFIDF_2000 | Brute |        51       | 0.5164948650860486 |
+------------+-------+-----------------+--------------------+
```

1. For all variants of KNN, our hyperparameter is same i.e 51.
2. AUC is Very high in case of BOW i.e 57%.
3. AUC is high & almost same in case of TFIDF & TFIDFW2V i.e 56%.
4. AUC is decreased by 5% for TFIDF when we took Top 2000 features instead of 50,000 Points.
5. So, overall BOW, TFIDF & TFIDFW2V are performing reasonable as compared to other variants of KNN.

In [ ]:

In [ ]:

In [ ]: