DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [2]:

```
# Run this cell to mount your Google Drive.
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6
qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%
b&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.
2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fww
```

```
ogleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code

Enter your authorization code:
··········
Mounted at /content/drive
```

In [3]:

```
%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 [0]:

```
project_data = pd.read_csv('./drive/My Drive/Colab Notebooks/train_data.csv')
resource_data = pd.read_csv('./drive/My Drive/Colab Notebooks/resources.csv')
```

In [5]:

```
 print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.head(2)
```

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

Out[5]:

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

In [6]:

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

Out[6]:

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

In [7]:

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


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


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


project_data.head(2)
```

Out[7]:

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

In [8]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[8]:

	id	description	quantity	price

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

1.2 preprocessing of `project_subject_categories`

In [0]:

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

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

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

from collections import Counter
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 [0]:

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

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

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

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

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

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

Preprocessing of Project_grade_category

In [11]:

```
grade_catogories = list(project_data['project_grade_category'].values)

grade_cat_list = []
for i in grade_catogories:
    temp = ""
    i = i.replace(' ','')
    temp = i.replace('-','_')
    grade_cat_list.append(temp.strip())

project_data['clean_gradecategories'] = grade_cat_list
project_data.drop(['project_grade_category'], 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_gradecategories'].values:
    my_counter.update(word.split())

grade_cat_dict = dict(my_counter)
sorted_grade_cat_dict = dict(sorted(grade_cat_dict.items(), key=lambda kv: kv[1]))
print(grade_cat_dict)
```

```
{'GradesPreK_2': 44225, 'Grades3_5': 37137, 'Grades9_12': 10963, 'Grades6_8': 16923}
```

Preprocessing of teacher_prefix

In [12]:

```
project_data['teacher_prefix']=project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix'].value_counts()
```

Out[12]:

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

1.3 Text preprocessing

In [0]:

```
# 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(2)
```

Out[14]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	project_essay_1	pr
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	I have been fortunate enough to use the Fairy ...	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms	UT	2016-04-27 00:31:25	Sensory Tools for Focus	Imagine being 8-9 years old. You're in your th...	

In [0]:

```
#### 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)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom as well as the STEM j
ournals, which my students really enjoyed.  I would love to implement more of the Lakeshore STEM k
its 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 statu
s.  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 projects.  I would use the kits and robot to help guide my science i
nstruction in engaging and meaningful ways.  I can adapt the kits to my current language arts paci
ng guide where we already teach some of the material in the kits like tall tales (Paul Bunyan) or
Johnny Appleseed.  The following units will be taught in the next school year where I will
implement these kits: magnets, motion, sink vs. float, robots.  I often get to these units and don
't know If I am teaching the right way or using the right materials.    The kits will give me
additional ideas, strategies, and lessons to prepare my students in science.It is challenging to d
evelop high quality science activities.  These kits give me the materials I need to provide my
students with science activities that will go along with the curriculum in my classroom.  Although
I have some things (like magnets) in my classroom, I don't know how to use them effectively.  The
kits 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 disabilities. My students all
vary in their ability level. However, the ultimate goal is to increase all students literacy level
s. This includes their reading, writing, and communication levels.I teach a really dynamic group o
f students. However, my students face a lot of challenges. My students all live in poverty and in
a dangerous neighborhood. Despite these challenges, I have students who have the the desire to def
eat these challenges. My students all have learning disabilities and currently all are performing
below grade level. My students are visual learners and will benefit from a classroom that fulfills
their preferred learning style.The materials I am requesting will allow my students to be prepared
for the classroom with the necessary supplies.  Too often I am challenged with students who come t
o school unprepared for class due to economic challenges.  I want my students to be able to focus
on learning and not how they will be able to get school supplies.  The supplies will last all year
.  Students will be able to complete written assignments and maintain a classroom journal.  The ch
art paper will be used to make learning more visual in class and to create posters to aid students
in their learning.  The students have access to a classroom printer.  The toner will be used to pr
int student work that is completed on the classroom Chromebooks.I want to try and remove all barri
ers for the students learning and create opportunities for learning. One of the biggest barriers i
s the students not having 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 back...what do you remember about your
grandparents?  How amazing would it be to be able to flip through a book to see a day in their
lives?My second graders are voracious readers! They love to read both fiction and nonfiction books
.  Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elephant, and Mercy Watson.
```

```
They also love to read about insects, space and plants. My students are hungry bookworms! My stude
nts are eager to learn and read about the world around them. My kids love to be at school and are
like little sponges absorbing everything around them. Their parents work long hours and usually do
not see their children. My students are usually cared for by their grandparents or a family
friend. Most of my students do not have someone who speaks English at home. Thus it is difficult f
or my students to acquire language.Now think forward... wouldn't it mean a lot to your kids,
nieces or nephews or grandchildren, to be able to see a day in your life today 30 years from now?
Memories are so precious to us and being able to share these memories with future generations will
be a rewarding experience.  As part of our social studies curriculum, students will be learning ab
out changes over time.  Students will be studying photos to learn about how their community has ch
anged over time.  In particular, we will look at photos to study how the land, buildings,
clothing, and schools have changed over time.  As a culminating activity, my students will capture
a slice of their history and preserve it through scrap booking. Key important events in their
young lives will be documented with the date, location, and names.   Students will be using photos
from home and from school to create their second grade memories.   Their scrap books will preserve
their unique stories for future generations to enjoy.Your donation to this project will provide my
second graders with an opportunity to learn about social studies in a fun and creative manner.  Th
rough their scrapbooks, children will share their story with others and have a historical document
for the rest of their lives.
==================================================
```

In [0]:

```
# 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[2000])
print(sent)
print("="*50)
```

```
\"Creativity is intelligence having fun.\" --Albert Einstein. Our elementary library at Greenville
Elementary is anything but a quiet, hushed space. It is a place for collaboration and research. It
is a place for incorporating technology. It is a place for innovation. And it is a place for creat
ing.Our school serves 350 third and fourth graders who primarily live in rural and poverty-stricke
n areas in our community. Being a Title I school, approximately 85% of them receive free or
reduced lunch. But they are inquisitive, creative, and eager to learn. They love visiting the libr
ary to check out books, hear \r\nstories, create digital stories, and use the computer lab for lea
rning and fun. We want to build our library is Makerspace with activities revolving around art and
literacy to provide more engaging, hands-on activities.We want to begin \"Makerspace Fridays!\"
Our school recently received a $1000 grant for books for our arts-integrated Makerspace. We have r
eceived titles such as \"Origami for Everyone,\" \"How to Make Stuff with Ducktape,\" and \"Cool E
ngineering Activities for Girls.\"  We now need supplies to correlate with these new informational
texts. By adding these art and craft supplies, students will be able to design and create
masterpieces related to their coursework. \r\n\r\nFor example, while studying Native Americans, st
udents can use the looms and yarn to recreate Navajo and/or Pueblo weaving. Weaving can also be in
tegrated with literacy through Greek mythology and the story of Arachne.\r\n\r\nCreating art with
perler beads has many possibilities! Students can design their own animals after studying their ch
aracteristics. They can use symmetry and patterning to create one-of-a-kind originals.
\r\n\r\nOrigami reinforces geometry, thinking skills, fractions, problem-solving, and just fun sci
ence!Our students need to be able to apply what they read and learn. If they read a how-to book, t
hey will apply that reading through a hands-on art activity and actually create a product. This is
a crucial skill in the real world. By creating and designing their own masterpieces, they are usin
g many critical thinking skills. Students will become more analytical thinkers.
==================================================
```

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)
```

```
 Creativity is intelligence having fun.  --Albert Einstein. Our elementary library at Greenville
Elementary is anything but a quiet, hushed space. It is a place for collaboration and research. It
is a place for incorporating technology. It is a place for innovation. And it is a place for creat
ing.Our school serves 350 third and fourth graders who primarily live in rural and poverty-stricke
n areas in our community. Being a Title I school, approximately 85% of them receive free or
reduced lunch. But they are inquisitive, creative, and eager to learn. They love visiting the libr
ary to check out books, hear   stories, create digital stories, and use the computer lab for learn
ing and fun. We want to build our library is Makerspace with activities revolving around art and l
iteracy to provide more engaging, hands-on activities.We want to begin  Makerspace Fridays!  Our s
chool recently received a $1000 grant for books for our arts-integrated Makerspace. We have receiv
ed titles such as  Origami for Everyone,   How to Make Stuff with Ducktape,  and  Cool Engineering
Activities for Girls.   We now need supplies to correlate with these new informational texts. By a
dding these art and craft supplies, students will be able to design and create masterpieces relate
d to their coursework.     For example, while studying Native Americans, students can use the loom
s and yarn to recreate Navajo and/or Pueblo weaving. Weaving can also be integrated with literacy
through Greek mythology and the story of Arachne.    Creating art with perler beads has many possi
bilities! Students can design their own animals after studying their characteristics. They can use
symmetry and patterning to create one-of-a-kind originals.     Origami reinforces geometry,
thinking skills, fractions, problem-solving, and just fun science!Our students need to be able to
apply what they read and learn. If they read a how-to book, they will apply that reading through a
hands-on art activity and actually create a product. This is a crucial skill in the real world. By
creating and designing their own masterpieces, they are using many critical thinking skills. Stude
nts will become more analytical thinkers.
```

In [20]:

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

```
 Creativity is intelligence having fun Albert Einstein Our elementary library at Greenville
Elementary is anything but a quiet hushed space It is a place for collaboration and research It is
a place for incorporating technology It is a place for innovation And it is a place for creating O
ur school serves 350 third and fourth graders who primarily live in rural and poverty stricken are
as in our community Being a Title I school approximately 85 of them receive free or reduced lunch
But they are inquisitive creative and eager to learn They love visiting the library to check out b
ooks hear stories create digital stories and use the computer lab for learning and fun We want to
build our library is Makerspace with activities revolving around art and literacy to provide more
engaging hands on activities We want to begin Makerspace Fridays Our school recently received a 10
00 grant for books for our arts integrated Makerspace We have received titles such as Origami for
Everyone How to Make Stuff with Ducktape and Cool Engineering Activities for Girls We now need sup
plies to correlate with these new informational texts By adding these art and craft supplies
students will be able to design and create masterpieces related to their coursework For example wh
ile studying Native Americans students can use the looms and yarn to recreate Navajo and or Pueblo
weaving Weaving can also be integrated with literacy through Greek mythology and the story of Arac
hne Creating art with perler beads has many possibilities Students can design their own animals af
ter studying their characteristics They can use symmetry and patterning to create one of a kind or
iginals Origami reinforces geometry thinking skills fractions problem solving and just fun science
Our students need to be able to apply what they read and learn If they read a how to book they wil
l apply that reading through a hands on art activity and actually create a product This is a cruci
al skill in the real world By creating and designing their own masterpieces they are using many cr
itical thinking skills Students will become more analytical thinkers
```

In [0]:

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

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

In [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, 1650.69it/s]
```

In [23]:

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

Out[23]:

```
'creativity intelligence fun albert einstein elementary library greenville elementary anything qui
et hushed space place collaboration research place incorporating technology place innovation place
creating school serves 350 third fourth graders primarily live rural poverty stricken areas
community title school approximately 85 receive free reduced lunch inquisitive creative eager
learn love visiting library check books hear stories create digital stories use computer lab learn
ing fun want build library makerspace activities revolving around art literacy provide engaging ha
nds activities want begin makerspace fridays school recently received 1000 grant books arts
integrated makerspace received titles origami everyone make stuff ducktape cool engineering activi
ties girls need supplies correlate new informational texts adding art craft supplies students able
design create masterpieces related coursework example studying native americans students use looms
yarn recreate navajo pueblo weaving weaving also integrated literacy greek mythology story arachne
creating art perler beads many possibilities students design animals studying characteristics use
symmetry patterning create one kind originals origami reinforces geometry thinking skills
fractions problem solving fun science students need able apply read learn read book apply reading
hands art activity actually create product crucial skill real world creating designing
masterpieces using many critical thinking skills students become analytical thinkers'
```

1.4 Preprocessing of `project_title`

In [24]:

```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_titles=[]
for sentance in tqdm(project_data['project_title'].values):
  sent=decontracted(sentance)
  sent=sent.replace('\\r',' ')
```

```
    sent=sent.replace('\\"',' ')
    sent=sent.replace('\\n',' ')
    sent=re.sub('[^A-za-z0-9]+',' ',sent)
    sent=' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:03<00:00, 34422.89it/s]
```

In [25]:

```
#After preprocessing titles
print(len(preprocessed_titles))
```

```
109248
```

In [26]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[26]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [0]:

```
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

1.5 Preparing data for models

In [28]:

```
project_data.columns
```

Out[28]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_title', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'clean_gradecategories',
       'essay', 'price', 'quantity'],
      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 value
   - Find the best hyper paramter using k-fold cross validation (or) simple cross validation data
   - Use gridsearch-cv or randomsearch-cv or write your own for loops to do this task
3. **Representation of results**
   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure
   - Once you find the best hyper parameter, you need to train your model-M using the best hyper-param. Now, find the AUC on test data and plot the ROC curve on both train and test using model-M.
   - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points
4. **[Task-2]**
   - Select top 2000 features from feature Set 2 using `SelectKBest` and then apply KNN on top of these features
     - ```
       from sklearn.datasets import load_digits
       from sklearn.feature_selection import SelectKBest, chi2
       X, y = load_digits(return_X_y=True)
       X.shape
       X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
       X_new.shape
       ========
       output:
       (1797, 64)
       (1797, 20)
       ```
   - Repeat the steps 2 and 3 on the data matrix after feature selection
5. **Conclusion**
   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

Note: Data Leakage

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

2. K Nearest Neighbor

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

In [0]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
```
```

```
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [0]:

```
project_data['preprocessed_essays']=preprocessed_essays
project_data['preprocessed_titles']=preprocessed_titles
```

In [0]:

```
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','project_essay_4'],axis=1
,inplace=True)
```

In [31]:

```
project_data=project_data[0:20000]
project_data.shape
```

Out[31]:

```
(20000, 18)
```

In [32]:

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

Out[32]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	project_resource_summa
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	My students need STEM k to learn critical s
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms	UT	2016-04-27 00:31:25	Sensory Tools for Focus	My students need Boo Boards for quiet sens

In [33]:

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

Out[33]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_title	project_resource_summa
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs	CA	2016-04-27 00:27:36	Engineering STEAM into the Primary Classroom	My students need STEM k to learn critical s

In [0]:

```
# Splitting the data
```

```
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)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [0]:

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

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

Encoding categorical features : clean_categories

In [36]:

```
# Encoding Categorical features :clean_categories
from sklearn.feature_extraction.text import CountVectorizer

vectorizer=CountVectorizer(vocabulary=list(cat_dict.keys()),lowercase=False,binary=True)
vectorizer.fit(X_train['clean_categories'].values)

X_train_clean_categories_ohe=vectorizer.transform(X_train['clean_categories'].values)
X_test_clean_categories_ohe=vectorizer.transform(X_test['clean_categories'].values)

print("After Vectorizations of Clean Categories")
print(X_train_clean_categories_ohe.shape,y_train.shape)
print(X_test_clean_categories_ohe.shape,y_test.shape)

print(vectorizer.get_feature_names())
```

```
After Vectorizations of Clean Categories
(13400, 9) (13400,)
(6600, 9) (6600,)
['Math_Science', 'SpecialNeeds', 'Literacy_Language', 'AppliedLearning', 'History_Civics',
'Music_Arts', 'Health_Sports', 'Warmth', 'Care_Hunger']
```

Encoding Categorical features :clean_subcategories

In [37]:

```
# Encoding Categorical features :clean_subcategories

vectorizer=CountVectorizer(vocabulary=list(sub_cat_dict.keys()),lowercase=False,binary=True)
vectorizer.fit(X_train['clean_subcategories'].values)

X_train_clean_subcategories_ohe=vectorizer.transform(X_train['clean_subcategories'].values)
X_test_clean_subcategories_ohe=vectorizer.transform(X_test['clean_subcategories'].values)

print("After Vectorizations")
print(X_train_clean_subcategories_ohe.shape,y_train.shape)
print(X_test_clean_subcategories_ohe.shape,y_test.shape)

print(vectorizer.get_feature_names())
```

```
After Vectorizations
(13400, 30) (13400,)
(6600, 30) (6600,)
['AppliedSciences', 'Health_LifeScience', 'SpecialNeeds', 'Literacy', 'EarlyDevelopment',
'Mathematics', 'SocialSciences', 'History_Geography', 'ESL', 'Extracurricular', 'VisualArts',
'EnvironmentalScience', 'Literature_Writing', 'Gym_Fitness', 'Music', 'TeamSports',
'PerformingArts', 'College_CareerPrep', 'Other', 'CharacterEducation', 'ForeignLanguages', 'Health
_Wellness', 'Civics_Government', 'Economics', 'CommunityService', 'FinancialLiteracy'
```

```
'Wellness', 'Civics_Government', 'Economics', 'CommunityService', 'FinancialLiteracy',
'NutritionEducation', 'ParentInvolvement', 'Warmth', 'Care_Hunger']
```

Encoding categorical features:teacher_prefix

In [38]:

```
# Encoding categorical features:teacher_prefix
from sklearn.feature_extraction.text import CountVectorizer

prefix_dict=dict(project_data['teacher_prefix'].value_counts())
print(prefix_dict.keys())

vectorizer=CountVectorizer(vocabulary=list(prefix_dict.keys()),lowercase=False,binary=True)
vectorizer.fit(X_train['teacher_prefix'].values.astype('U'))

X_train_teacher_prefix_ohe=vectorizer.transform(X_train['teacher_prefix'].values.astype('U'))
X_test_teacher_prefix_ohe=vectorizer.transform(X_test['teacher_prefix'].values.astype('U'))

print("After Vectorizations of teacher_prefix")
print(X_train_teacher_prefix_ohe.shape,y_train.shape)
print(X_test_teacher_prefix_ohe.shape,y_test.shape)

print(vectorizer.get_feature_names())
```

```
dict_keys(['Mrs', 'Ms', 'Mr', 'Teacher', 'Dr'])
After Vectorizations of teacher_prefix
(13400, 5) (13400,)
(6600, 5) (6600,)
['Mrs', 'Ms', 'Mr', 'Teacher', 'Dr']
```

Encoding categorical features:School_state

In [39]:

```
#Encoding Categorical Features:school_state

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


state_dict = dict(my_counter)
sorted_state_dict = dict(sorted(state_dict.items(), key=lambda kv: kv[1]))
print(sorted_state_dict)


vectorizer=CountVectorizer(vocabulary=list(state_dict.keys()),lowercase=False,binary=True)
vectorizer.fit(X_train['school_state'].values)

X_train_state_ohe=vectorizer.transform(X_train['school_state'].values)
X_test_state_ohe=vectorizer.transform(X_test['school_state'].values)

print("After Vectorization of school_state")
print(X_train_state_ohe.shape,y_train.shape)
print(X_test.shape,y_test.shape)
print(vectorizer.get_feature_names())
```

```
{'WY': 9, 'ND': 21, 'VT': 23, 'MT': 31, 'AK': 39, 'RI': 47, 'NH': 58, 'SD': 59, 'NE': 70, 'ME': 81
, 'HI': 83, 'DE': 88, 'ID': 95, 'DC': 111, 'WV': 129, 'KS': 139, 'IA': 141, 'NM': 150, 'OR': 186,
'MN': 201, 'WI': 211, 'UT': 213, 'AR': 217, 'CO': 235, 'NV': 238, 'CT': 254, 'MS': 263, 'KY': 268,
'AL': 278, 'MD': 342, 'LA': 345, 'NJ': 360, 'MA': 372, 'VA': 382, 'TN': 407, 'WA': 408, 'MI': 463,
'AZ': 469, 'OK': 476, 'OH': 513, 'MO': 551, 'PA': 598, 'IN': 615, 'SC': 731, 'IL': 802, 'GA': 828,
'NC': 1005, 'FL': 1293, 'NY': 1299, 'TX': 1534, 'CA': 2269}
After Vectorization of school_state
(13400, 51) (13400,)
(6600, 17) (6600,)
['CA', 'UT', 'GA', 'WA', 'HI', 'IL', 'OH', 'KY', 'SC', 'FL', 'MO', 'MI', 'NY', 'VA', 'MD', 'TX', 'M
S', 'NJ', 'AZ', 'OK', 'PA', 'WV', 'NC', 'CO', 'DC', 'MA', 'ID', 'AL', 'ME', 'TN', 'IN', 'LA', 'CT',
'AR', 'KS', 'OR', 'WI', 'IA', 'SD', 'AK', 'MN', 'NM', 'NV', 'MT', 'RI', 'NH', 'WY', 'NE', 'DE', 'NI
', 'VT']
```

Encoding categorical features:project_grade_category

In [40]:

```
#Encoding Categorical Features:project_grade_category
vectorizer=CountVectorizer(vocabulary=list(grade_cat_dict.keys()),lowercase=False,binary=True)
vectorizer.fit(X_train['clean_gradecategories'].values)

X_train_grade_ohe=vectorizer.transform(X_train['clean_gradecategories'].values)
X_test_grade_ohe=vectorizer.transform(X_test['clean_gradecategories'].values)

print("After Vectorizations of project_grade_category")
print(X_train_grade_ohe.shape,y_train.shape)
print(X_test_grade_ohe.shape,y_test.shape)
print(vectorizer.get_feature_names())
print(X_test_grade_ohe.toarray())
```

```
After Vectorizations of project_grade_category
(13400, 4) (13400,)
(6600, 4) (6600,)
['GradesPreK_2', 'Grades3_5', 'Grades9_12', 'Grades6_8']
[[0 0 0 1]
 [0 1 0 0]
 [0 1 0 0]
 ...
 [0 0 0 1]
 [0 0 0 1]
 [0 1 0 0]]
```

Encoding Numerical Feature: Price

In [41]:

```
# Encoding Numerical Feature: Price
from sklearn.preprocessing import Normalizer
normalizer=Normalizer()
normalizer.fit(X_train['price'].values.reshape(-1,1))
X_train_price_norm=normalizer.transform(X_train['price'].values.reshape(-1,1))
X_test_price_norm=normalizer.transform(X_test['price'].values.reshape(-1,1))

print("After Vectorizations of price")
print(X_train_price_norm.shape,y_train.shape)
print(X_test_price_norm.shape,y_test.shape)
```

```
After Vectorizations of price
(13400, 1) (13400,)
(6600, 1) (6600,)
```

Encoding Numerical Feature :teacher_number_of_previously_posted_projects

In [42]:

```
#Encoding Numerical Feature :teacher_number_of_previously_posted_projects

from sklearn.preprocessing import Normalizer
normalizer=Normalizer()
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_train_teacher_ppp_norm=normalizer.transform(X_train['teacher_number_of_previously_posted_projects
].values.reshape(-1,1))
X_test_teacher_ppp_norm=normalizer.transform(X_test['teacher_number_of_previously_posted_projects'
].values.reshape(-1,1))

print("After Vectorizations of price")
print(X_train_teacher_ppp_norm.shape,y_train.shape)
print(X_test_teacher_ppp_norm.shape,y_test.shape)
```

```
After Vectorizations of price
(13400, 1) (13400,)
(6600, 1) (6600,)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [0]:

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

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

Vectorization using BOW

In [44]:

```
# Encoding preprocessed_essay for  BOW
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['preprocessed_essays'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_preprocessed_essay_bow = vectorizer.transform(X_train['preprocessed_essays'].values)
X_test_preprocessed_essay_bow = vectorizer.transform(X_test['preprocessed_essays'].values)

print("After vectorizations of preprocessed_essay")
print(X_train_preprocessed_essay_bow.shape, y_train.shape)
print(X_test_preprocessed_essay_bow.shape, y_test.shape)
```

```
After vectorizations of preprocessed_essay
(13400, 5000) (13400,)
(6600, 5000) (6600,)
```

In [45]:

```
# Encoding preprocessed_titles for BOW
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['preprocessed_titles'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_preprocessed_title_bow = vectorizer.transform(X_train['preprocessed_titles'].values)
X_test_preprocessed_title_bow = vectorizer.transform(X_test['preprocessed_titles'].values)

print("After vectorizations of preprocessed_titles")
print(X_train_preprocessed_title_bow.shape, y_train.shape)
print(X_test_preprocessed_title_bow.shape, y_test.shape)
```

```
After vectorizations of preprocessed_titles
(13400, 1051) (13400,)
(6600, 1051) (6600,)
```

Vectorization using Tf-idf

In [46]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer=TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['preprocessed_essays'].values)

X_train_preprocessed_essay_tfidf=vectorizer.transform(X_train['preprocessed_essays'].values)
X_test_preprocessed_essay_tfidf=vectorizer.transform(X_test['preprocessed_essays'].values)

print("After tfidf vectorization of preprocessed_essays")
```

```
print(X_train_preprocessed_essay_tfidf.shape,y_train.shape)
print(X_test_preprocessed_essay_tfidf.shape,y_test.shape)
```

```
After tfidf vectorization of preprocessed_essays
(13400, 5000) (13400,)
(6600, 5000) (6600,)
```

In [47]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer=TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['preprocessed_titles'].values)

X_train_preprocessed_title_tfidf=vectorizer.transform(X_train['preprocessed_titles'].values)
X_test_preprocessed_title_tfidf=vectorizer.transform(X_test['preprocessed_titles'].values)

print("After tfidf vectorization of preprocessed_title")
print(X_train_preprocessed_title_tfidf.shape,y_train.shape)
print(X_test_preprocessed_title_tfidf.shape,y_test.shape)
```

```
After tfidf vectorization of preprocessed_title
(13400, 1051) (13400,)
(6600, 1051) (6600,)
```

Vectorization using avg-W2V

In [0]:

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

In [49]:

```
# Avg-W2V on Preprocessed_essays
X_train_preprocessed_essay_avg_w2v_vectors=[]
for sentence in tqdm(X_train['preprocessed_essays']):
  vector=np.zeros(300)
  cnt_words=0
  for word in sentence.split():
    if word in glove_words:
      vector+=model[word]
      cnt_words+=1
  if cnt_words!=0:
    vector/=cnt_words
  X_train_preprocessed_essay_avg_w2v_vectors.append(vector)
print(len(X_train_preprocessed_essay_avg_w2v_vectors))
print(len(X_train_preprocessed_essay_avg_w2v_vectors[0]))
print("*"*50)

X_test_preprocessed_essay_avg_w2v_vectors=[]
for sentence in tqdm(X_test['preprocessed_essays']):
  vector=np.zeros(300)
  cnt_words=0
  for word in sentence.split():
    if word in glove_words:
      vector+=model[word]
      cnt_words+=1
  if cnt_words!=0:
    vector/=cnt_words
  X_test_preprocessed_essay_avg_w2v_vectors.append(vector)
print(len(X_test_preprocessed_essay_avg_w2v_vectors))
print(len(X_test_preprocessed_essay_avg_w2v_vectors[0]))
print("*"*50)
```

```
100%|██████████| 13400/13400 [00:04<00:00, 3176.57it/s]
  5%|▌         | 333/6600 [00:00<00:01, 3324.04it/s]
```

```
13400
300
**************************************************
```

```
100%|██████████| 6600/6600 [00:02<00:00, 3227.33it/s]
```

```
6600
300
**************************************************
```

In [50]:

```
''# Avg-W2V on Preprocessed_titles
X_train_preprocessed_title_avg_w2v_vectors=[]
for sentence in tqdm(X_train['preprocessed_titles']):
  vector=np.zeros(300)
  cnt_words=0
  for word in sentence.split():
    if word in glove_words:
      vector+=model[word]
      cnt_words+=1
  if cnt_words!=0:
    vector/=cnt_words
  X_train_preprocessed_title_avg_w2v_vectors.append(vector)
print(len(X_train_preprocessed_title_avg_w2v_vectors))
print(len(X_train_preprocessed_title_avg_w2v_vectors[0]))
print("*"*50)


X_test_preprocessed_title_avg_w2v_vectors=[]
for sentence in tqdm(X_test['preprocessed_titles']):
  vector=np.zeros(300)
  cnt_words=0
  for word in sentence.split():
    if word in glove_words:
      vector+=model[word]
      cnt_words+=1
  if cnt_words!=0:
    vector/=cnt_words
  X_test_preprocessed_title_avg_w2v_vectors.append(vector)
print(len(X_test_preprocessed_title_avg_w2v_vectors))
print(len(X_test_preprocessed_title_avg_w2v_vectors[0]))
print("*"*50)
```

```
100%|██████████| 13400/13400 [00:00<00:00, 68763.29it/s]
100%|██████████| 6600/6600 [00:00<00:00, 69889.36it/s]
```

```
13400
300
**************************************************
6600
300
**************************************************
```

Vectorization using Tfidf weighted W2V

In [0]:

```
# Tfidf for Preprocessed_essays
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_essays'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_X_train_preprocessed_essays_words = set(tfidf_model.get_feature_names())
```

In [52]:

```
X_train_preprocessed_essay_tfidf_w2v_vectors=[]
for sentence in tqdm(X_train['preprocessed_essays']):
  vector=np.zeros(300)
  tf_idf_weight=0
  for word in sentence.split():
    if (word in glove_words) and (word in tfidf_X_train_preprocessed_essays_words):
      vec=model[word]
      tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
      vector += (vec * tf_idf)
      tf_idf_weight += tf_idf
  if tf_idf_weight!=0:
    vector/=tf_idf_weight
  X_train_preprocessed_essay_tfidf_w2v_vectors.append(vector)
print(len(X_train_preprocessed_essay_tfidf_w2v_vectors))
print(len(X_train_preprocessed_essay_tfidf_w2v_vectors[0]))
print("*"*50)

X_test_preprocessed_essay_tfidf_w2v_vectors=[]
for sentence in tqdm(X_test['preprocessed_essays']):
  vector=np.zeros(300)
  tf_idf_weight=0
  for word in sentence.split():
    if (word in glove_words) and (word in tfidf_X_train_preprocessed_essays_words):
      vec=model[word]
      tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
      vector += (vec * tf_idf)
      tf_idf_weight += tf_idf
  if tf_idf_weight!=0:
    vector/=tf_idf_weight
  X_test_preprocessed_essay_tfidf_w2v_vectors.append(vector)
print(len(X_test_preprocessed_essay_tfidf_w2v_vectors))
print(len(X_test_preprocessed_essay_tfidf_w2v_vectors[0]))
print("*"*50)
```

```
100%|██████████| 13400/13400 [00:24<00:00, 542.35it/s]
  1%|          | 56/6600 [00:00<00:11, 550.33it/s]
```

```
13400
300
**************************************************
```

```
100%|██████████| 6600/6600 [00:12<00:00, 542.19it/s]
```

```
6600
300
**************************************************
```

In [0]:

```
# Tfidf for Preprocessed_titles
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_titles'])
# 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_X_train_preprocessed_titles_words = set(tfidf_model.get_feature_names())
```

In [54]:

```
X_train_preprocessed_title_tfidf_w2v_vectors=[]
for sentence in tqdm(X_train['preprocessed_titles']):
  vector=np.zeros(300)
  tf_idf_weight=0
  for word in sentence.split():
    if (word in glove_words) and (word in tfidf_X_train_preprocessed_titles_words):
      vec=model[word]
      tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
      vector += (vec * tf_idf)
      tf_idf_weight += tf_idf
  if tf_idf_weight!=0:
    vector/=tf_idf_weight
```

```
  X_train_preprocessed_title_tfidf_w2v_vectors.append(vector)
print(len(X_train_preprocessed_title_tfidf_w2v_vectors))
print(len(X_train_preprocessed_title_tfidf_w2v_vectors[0]))
print("*"*50)


X_test_preprocessed_title_tfidf_w2v_vectors=[]
for sentence in tqdm(X_test['preprocessed_titles']):
  vector=np.zeros(300)
  tf_idf_weight=0
  for word in sentence.split():
    if (word in glove_words) and (word in tfidf_X_train_preprocessed_titles_words):
      vec=model[word]
      tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
      vector += (vec * tf_idf)
      tf_idf_weight += tf_idf
  if tf_idf_weight!=0:
    vector/=tf_idf_weight
  X_test_preprocessed_title_tfidf_w2v_vectors.append(vector)
print(len(X_test_preprocessed_title_tfidf_w2v_vectors))
print(len(X_test_preprocessed_title_tfidf_w2v_vectors[0]))
print("*"*50)
```

```
100%|██████████| 13400/13400 [00:00<00:00, 25952.31it/s]
 34%|███       | 2267/6600 [00:00<00:00, 22663.43it/s]
```

```
13400
300
**************************************************
```

```
100%|██████████| 6600/6600 [00:00<00:00, 27410.47it/s]
```

```
6600
300
**************************************************
```

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

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

In [0]:

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

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

2.4.1 Applying KNN brute force on BOW, SET 1

In [56]:

```
from scipy.sparse import hstack
X_train_1=hstack((X_train_clean_categories_ohe,X_train_clean_subcategories_ohe,X_train_teacher_prefi
ix_ohe,X_train_state_ohe,X_train_grade_ohe,X_train_price_norm,X_train_teacher_ppp_norm,X_train_prep
rocessed_essay_bow,X_train_preprocessed_title_bow)).tocsr()
X_test_1=hstack((X_test_clean_categories_ohe,X_test_clean_subcategories_ohe,X_test_teacher_prefix_o
he,X_test_state_ohe,X_test_grade_ohe,X_test_price_norm,X_test_teacher_ppp_norm,X_test_preprocessed_
essay_bow,X_test_preprocessed_title_bow)).tocsr()
```

```
print("Final datamatrix for BOW")
print(X_train_1.shape,y_train.shape)
print(X_test_1.shape,y_test.shape)
```

```
Final datamatrix for BOW
(13400, 6152) (13400,)
(6600, 6152) (6600,)
```

In [0]:

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

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

    return y_data_pred
```

In [0]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1,5,21,33,45,65,75,87,91]}
clf = RandomizedSearchCV(neigh, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_1, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_n_neighbors'])

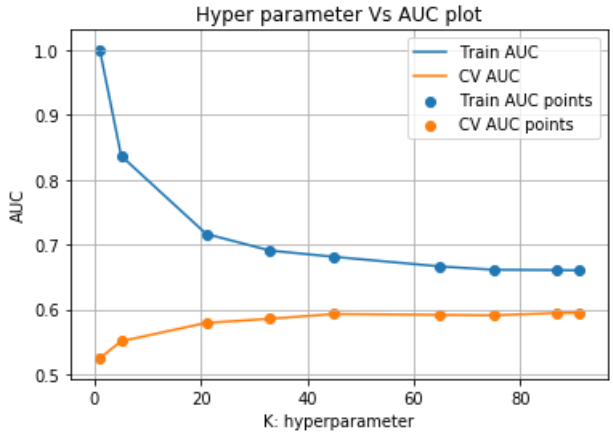
train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
K =  results['param_n_neighbors']

plt.plot(K, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
#plt.gca().fill_between(K, train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

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

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


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
print(clf.best_params_)
results.head(2)
```




```
{'n_neighbors': 91}
```

Out[0]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params	split0_test_score	split1_test_score
0	0.014962	0.000365	4.339754	0.041071	1	{'n_neighbors': 1}	0.529677	0.522370
1	0.015036	0.000120	4.573643	0.096627	5	{'n_neighbors': 5}	0.542317	0.544726

In [0]:

```
#gap between train auc and cv auc is less starts from 50
best_k=91
```

In [59]:

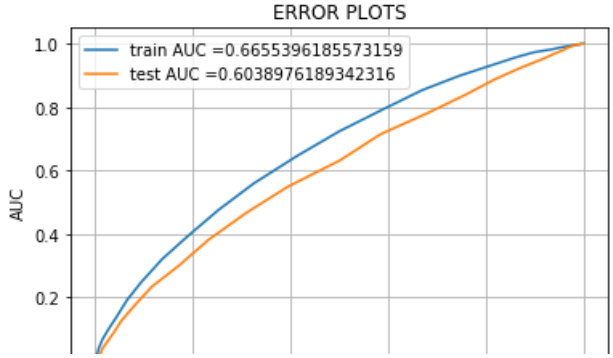
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

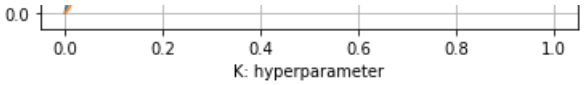
neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_train_1, y_train)

y_train_pred_1 = batch_predict(neigh, X_train_1)
y_test_pred_1 = batch_predict(neigh, X_test_1)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_1)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_1)

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





In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

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

In [61]:

```
best_train_1 = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
best_test_1 = find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
the maximum value of tpr*(1-fpr) 0.3789845796526433 for threshold 0.813
the maximum value of tpr*(1-fpr) 0.3324475741191038 for threshold 0.824
```

In [62]:

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

print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_1, best_train_1)))
print("Test confusion matrix")

print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_1, best_test_1)))
```

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

Train confusion matrix
[[1199  842]
 [4031 7328]]
Test confusion matrix
[[ 610  396]
 [2527 3067]]
```

BOW Train Confusion matrix

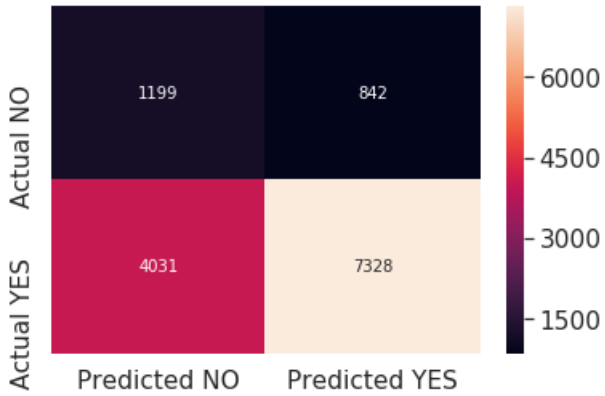
In [88]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
confusion_train=pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred_1,
best_train_1)),range(2),range(2))
confusion_train.columns = ['Predicted NO','Predicted YES']
confusion_train = confusion_train.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(confusion_train,annot=True,annot_kws={"size":10},fmt='g')
```

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

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc4178ecc0>
```



BOW Test Confusion matrix

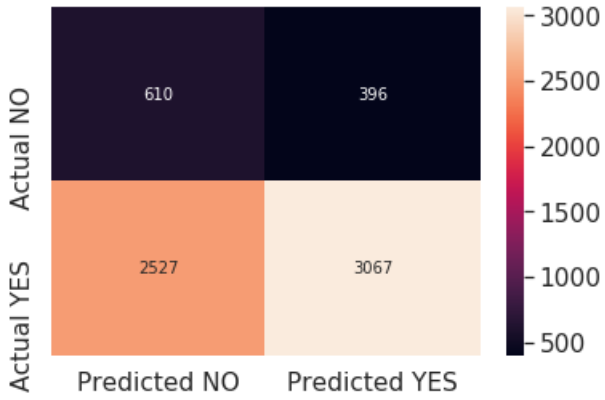
In [89]:

```
print("="*100)
confusion_test=pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred_1, best_test_1
)),range(2),range(2))
confusion_test.columns = ['Predicted NO','Predicted YES']
confusion_test = confusion_test.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.heatmap(confusion_test,annot=True,annot_kws={"size":10},fmt='g')
```

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

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc58a912b0>
```



2.4.2 Applying KNN brute force on TFIDF, SET 2

In [63]:

```
from scipy.sparse import hstack
X_train_2=hstack((X_train_clean_categories_ohe,X_train_clean_subcategories_ohe,X_train_teacher_prefi
ix_ohe,X_train_state_ohe,X_train_grade_ohe,X_train_price_norm,X_train_teacher_ppp_norm,X_train_prep
rocessed_essay_tfidf,X_train_preprocessed_title_tfidf)).tocsr()
X_test_2=hstack((X_test_clean_categories_ohe,X_test_clean_subcategories_ohe,X_test_teacher_prefix_c
he,X_test_state_ohe,X_test_grade_ohe,X_test_price_norm,X_test_teacher_ppp_norm,X_test_preprocessed_
essay_tfidf,X_test_preprocessed_title_tfidf)).tocsr()


print("After Vectorizations of tfidf")
print(X_train_2.shape,y_train.shape)
print(X_test_2.shape,y_test.shape)
```

```
After Vectorizations of tfidf
(13400, 6152) (13400,)
```

(15400, 6152) (15400,)
(6600, 6152) (6600,)

In [0]:

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

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

    return y_data_pred
```

In [0]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

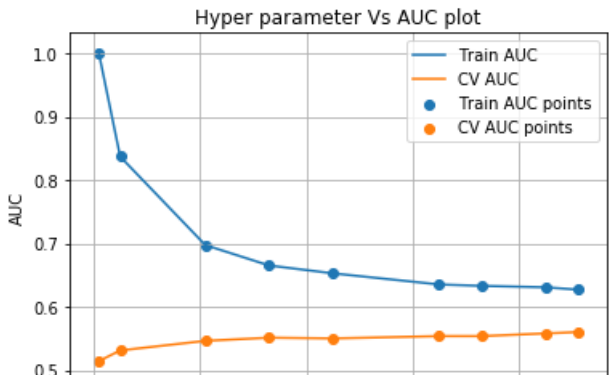
neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1,5,21,33,45,65,73,85,91]}
clf = RandomizedSearchCV(neigh, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_2, y_train)

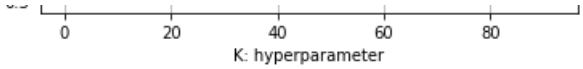
results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_n_neighbors'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
K =  results['param_n_neighbors']

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


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
print(clf.best_params_)
results.head(2)
```






```
{'n_neighbors': 91}
```

Out[0]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params	split0_test_score	split1_test_score
0	0.016382	0.002165	4.179279	0.273130	1	{'n_neighbors': 1}	0.514500	0.515053
1	0.013790	0.000283	4.697670	0.008125	5	{'n_neighbors': 5}	0.537966	0.527889

In [0]:

```
#gap between train auc and cv auc is less starts from 50
best_k=91
```

In [66]:

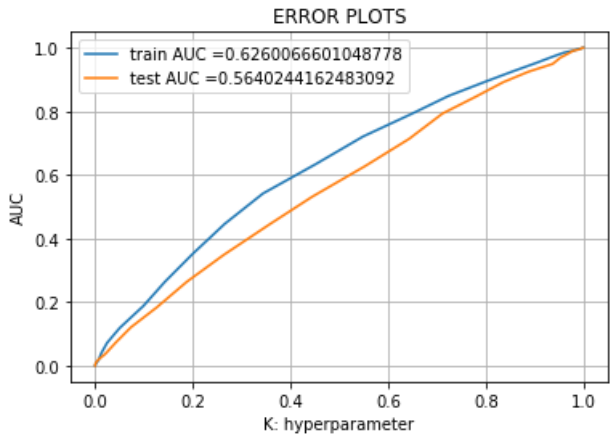
```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_train_2, y_train)

y_train_pred_2 = batch_predict(neigh, X_train_2)
y_test_pred_2 = batch_predict(neigh, X_test_2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_2)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_2)

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



In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t
```

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

In [68]:

```
best_train_2 = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
best_test_2 = find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
the maximum value of tpr*(1-fpr) 0.3554884787897922 for threshold 0.857
the maximum value of tpr*(1-fpr) 0.2952161894560418 for threshold 0.857
```

In [69]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_2, best_train_2)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_2, best_test_2)))
```

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

Train confusion matrix
[[1339  702]
 [5204 6155]]
Test confusion matrix
[[ 559  447]
 [2622 2972]]
```

Tfidf Train Confusion matrix

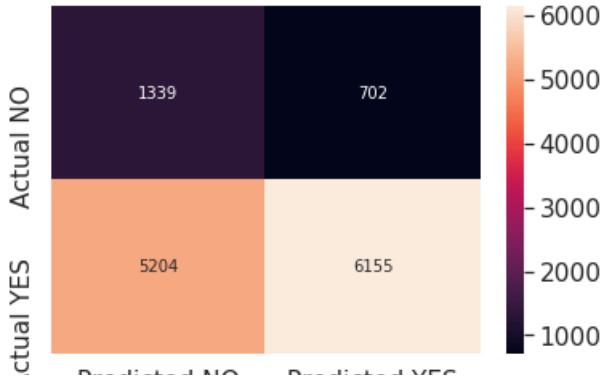
In [90]:

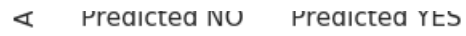
```
print("="*100)
from sklearn.metrics import confusion_matrix
confusion_train=pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred_2,
best_train_2)),range(2),range(2))
confusion_train.columns = ['Predicted NO','Predicted YES']
confusion_train = confusion_train.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(confusion_train,annot=True,annot_kws={"size":10},fmt='g')
```

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

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc425380b8>
```





Tfidf Test Confusion matrix

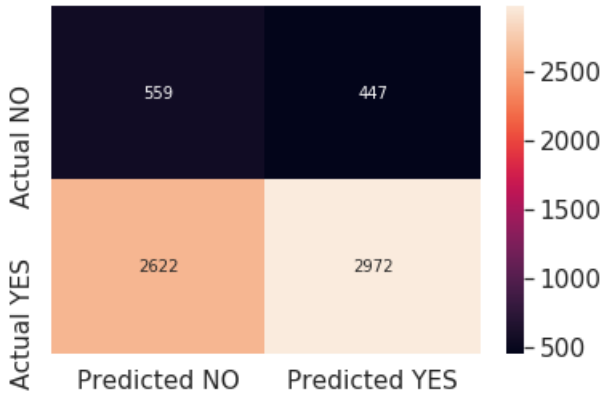
In [91]:

```
print("="*100)
confusion_test=pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred_2, best_test_2
)),range(2),range(2))
confusion_test.columns = ['Predicted NO','Predicted YES']
confusion_test = confusion_test.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.heatmap(confusion_test,annot=True,annot_kws={"size":10},fmt='g')
```

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

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc42534198>
```



2.4.3 Applying KNN brute force on AVG W2V, SET 3

In [0]:

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

In [70]:

```
from scipy.sparse import hstack
X_train_3=hstack((X_train_clean_categories_ohe,X_train_clean_subcategories_ohe,X_train_teacher_pref
ix_ohe,X_train_state_ohe,X_train_grade_ohe,X_train_price_norm,X_train_teacher_ppp_norm,X_train_prep
rocessed_essay_avg_w2v_vectors,X_train_preprocessed_title_avg_w2v_vectors)).tocsr()
X_test_3=hstack((X_test_clean_categories_ohe,X_test_clean_subcategories_ohe,X_test_teacher_prefix_o
he,X_test_state_ohe,X_test_grade_ohe,X_test_price_norm,X_test_teacher_ppp_norm,X_test_preprocessed_
essay_avg_w2v_vectors,X_test_preprocessed_title_avg_w2v_vectors)).tocsr()

print("After Vectorizations of avg w2v")
print(X_train_3.shape,y_train.shape)
print(X_test_3.shape,y_test.shape)
```

```
After Vectorizations of avg w2v
(13400, 701) (13400,)
(6600, 701) (6600,)
```

In [0]:

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

    y_data_pred = []
```

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

    return y_data_pred
```

In [0]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

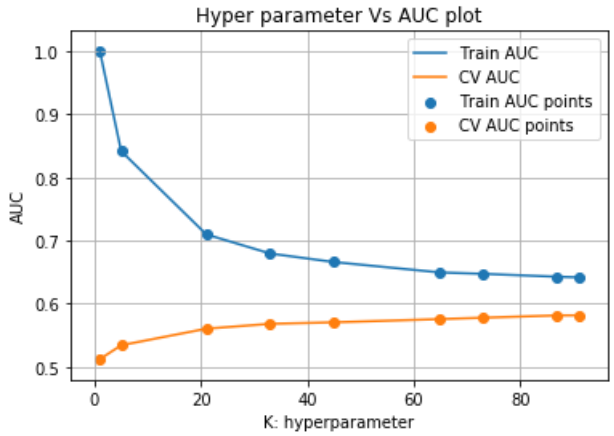
neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1,5,21,33,45,65,73,87,91]}
clf = RandomizedSearchCV(neigh, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_3, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_n_neighbors'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
K =  results['param_n_neighbors']

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


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
print(clf.best_params_)
results.head(2)
```




```
{'n_neighbors': 91}
```

Out[0]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params	split0_test_score	split1_test_score
0	0.074999	0.010653	65.760539	0.115644	1	{'n_neighbors':	0.509219	0.505609

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params	split0_test_score	split1_test_score
1	0.064299	0.000255	66.966410	0.564746	5	{'n_neighbors': 5}	0.536208	0.529695

In [72]:

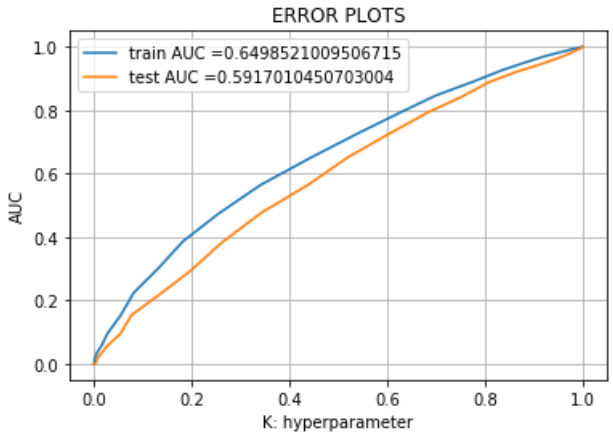
```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

best_k=91
neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_train_3, y_train)

y_train_pred_3 = batch_predict(neigh, X_train_3)
y_test_pred_3 = batch_predict(neigh, X_test_3)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_3)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_3)

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



In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

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

In [74]:

```
best_train_3 = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
best_test_3 = find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
the maximum value of tpr*(1-fpr) 0.3717452752080027 for threshold 0.857
the maximum value of tpr*(1-fpr) 0.3169989714910395 for threshold 0.857
```

In [75]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_3, best_train_3)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_3, best_test_3)))
```

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

Train confusion matrix
[[1346  695]
 [4956 6403]]
Test confusion matrix
[[ 564  442]
 [2431 3163]]
```

Avg W2V Train Confusion matrix

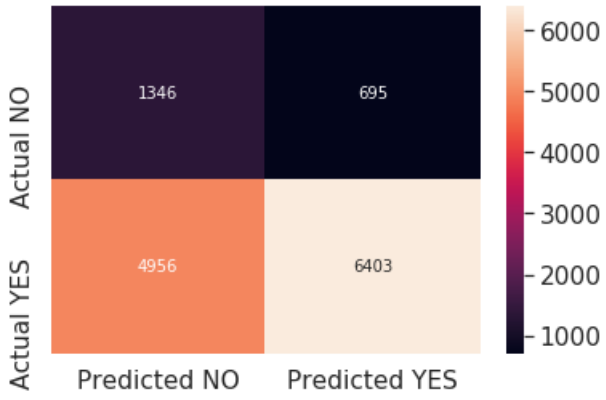
In [92]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
confusion_train=pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred_3,
best_train_3)),range(2),range(2))
confusion_train.columns = ['Predicted NO','Predicted YES']
confusion_train = confusion_train.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(confusion_train,annot=True,annot_kws={"size":10},fmt='g')
```

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

Out[92]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc41641898>
```



Avg W2V Test Confusion matrix

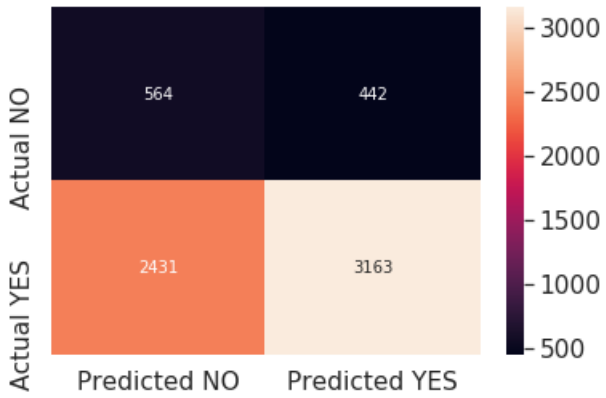
In [93]:

```
print("="*100)
confusion_test=pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred_3, best_test_3
)),range(2),range(2))
confusion_test.columns = ['Predicted NO','Predicted YES']
confusion_test = confusion_test.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.heatmap(confusion_test,annot=True,annot_kws={"size":10},fmt='g')
```

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

Out[93]:

<matplotlib.axes._subplots.AxesSubplot at 0x7efc41232c50>



2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [76]:

```
from scipy.sparse import hstack
X_train_4=hstack((X_train_clean_categories_ohe,X_train_clean_subcategories_ohe,X_train_teacher_prefix_ohe,X_train_state_ohe,X_train_grade_ohe,X_train_price_norm,X_train_teacher_ppp_norm,X_train_preprocessed_essay_tfidf_w2v_vectors,X_train_preprocessed_title_tfidf_w2v_vectors)).tocsr()
X_test_4=hstack((X_test_clean_categories_ohe,X_test_clean_subcategories_ohe,X_test_teacher_prefix_ohe,X_test_state_ohe,X_test_grade_ohe,X_test_price_norm,X_test_teacher_ppp_norm,X_test_preprocessed_essay_tfidf_w2v_vectors,X_test_preprocessed_title_tfidf_w2v_vectors)).tocsr()

print("After Vectorizations of Tfidf w2v")
print(X_train_4.shape,y_train.shape)
print(X_test_4.shape,y_test.shape)
```

```
After Vectorizations of Tfidf w2v
(13400, 701) (13400,)
(6600, 701) (6600,)
```

In [0]:

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

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

    return y_data_pred
```

In [0]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1,5,21,33,45,65,73,85,93]}
```

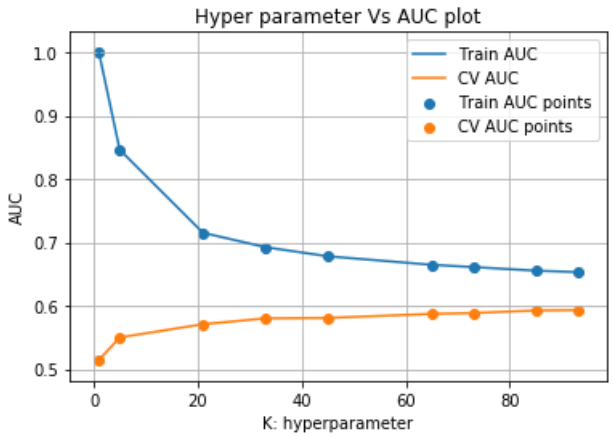
```
clf = RandomizedSearchCV(neigh, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_4, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_n_neighbors'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
K =  results['param_n_neighbors']

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


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
print(clf.best_params_)
results.head(2)
```




```
{'n_neighbors': 93}
```

Out[0]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params	split0_test_score	split1_test_score
0	0.078460	0.019157	66.490442	0.780343	1	{'n_neighbors': 1}	0.519376	0.511812
1	0.064624	0.000543	62.213221	0.620959	5	{'n_neighbors': 5}	0.556510	0.552063

In [78]:

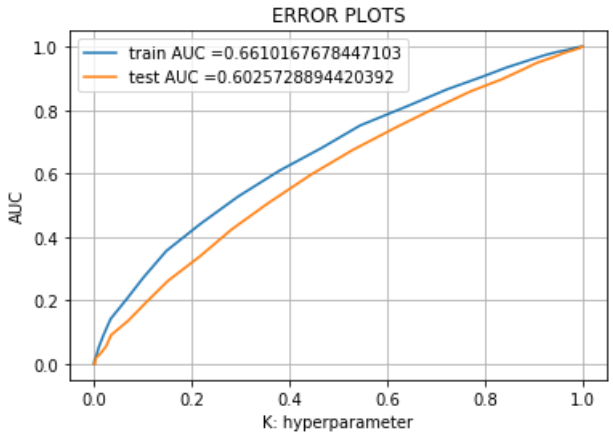
```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

best_k=93
neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_train_4, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred_4= batch_predict(neigh, X_train_4)
y_test_pred_4= batch_predict(neigh, X_test_4)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_4)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_4)
```

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



In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

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

In [80]:

```
best_train_4 = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
best_test_4= find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
the maximum value of tpr*(1-fpr) 0.3779623105335257 for threshold 0.849
the maximum value of tpr*(1-fpr) 0.33157366135684996 for threshold 0.849
```

In [81]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_4, best_train_4)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_4, best_test_4)))
```

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

the maximum value of tpr*(1-fpr) 0.3779623105335257 for threshold 0.849
Train confusion matrix
[[1267  774]
 [4443 6916]]
Test confusion matrix
[[ 558  448]
```

```
[2250 3344]]
```

Tfidf W2V Train Confusion matrix

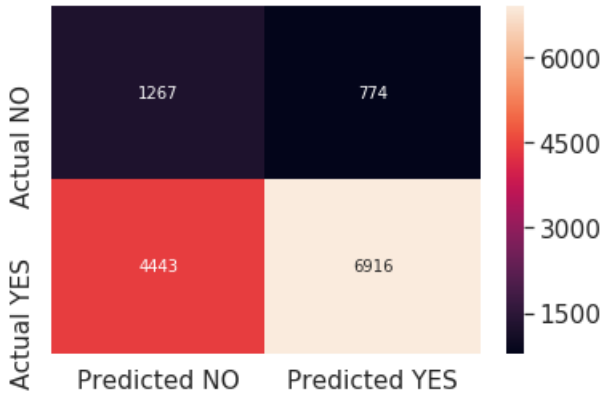
In [94]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
confusion_train=pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred_4,
best_train_4)),range(2),range(2))
confusion_train.columns = ['Predicted NO','Predicted YES']
confusion_train = confusion_train.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(confusion_train,annot=True,annot_kws={"size":10},fmt='g')
```

==========

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc42b36240>
```



Tfidf W2V Test Confusion matrix

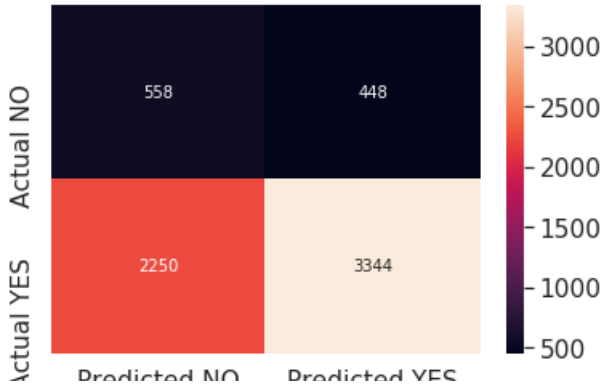
In [95]:

```
print("="*100)
confusion_test=pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred_4, best_test_4
)),range(2),range(2))
confusion_test.columns = ['Predicted NO','Predicted YES']
confusion_test = confusion_test.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.heatmap(confusion_test,annot=True,annot_kws={"size":10},fmt='g')
```

==========

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7efc41215c50>
```



Predicted NO Predicted YES

2.5 Feature selection with `SelectKBest`

In [0]:

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

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

In [82]:

```
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2
selector = SelectKBest(chi2, k=2000)
X_train_new=selector.fit_transform(X_train_2,y_train)
print(X_train_new.shape)
X_test_new=selector.transform(X_test_2)
print(X_test_new.shape)
```

```
(13400, 2000)
(6600, 2000)
```

In [0]:

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

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

    return y_data_pred
```

In [0]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score

neigh = KNeighborsClassifier(n_jobs=-1)
parameters = {'n_neighbors':[1,5,21,33,45,65,73,87,93]}
clf = RandomizedSearchCV(neigh, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X_train_new, y_train)

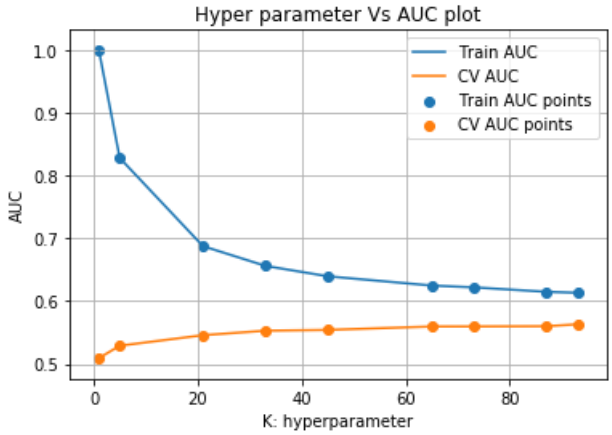
results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_n_neighbors'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
```

```
K =  results['param_n_neighbors']

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


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
print(clf.best_params_)
results.head(2)
```




```
{'n_neighbors': 93}
```

Out[0]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_n_neighbors	params	split0_test_score	split1_test_score
0	0.010817	0.004374	3.351677	0.106768	1	{'n_neighbors': 1}	0.513905	0.514147
1	0.007878	0.000084	3.630827	0.020355	5	{'n_neighbors': 5}	0.525094	0.546661

In [84]:

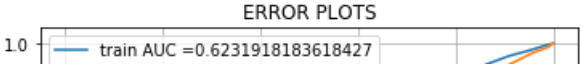
```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

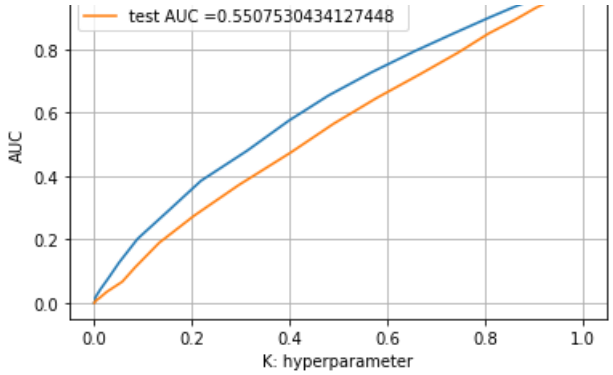
best_k=93
neigh = KNeighborsClassifier(n_neighbors=best_k, n_jobs=-1)
neigh.fit(X_train_new, y_train)

y_train_pred_best = batch_predict(neigh, X_train_new)
y_test_pred_best = batch_predict(neigh, X_test_new)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_best)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_best)

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





In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

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

In [86]:

```
best_train_best = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
best_test_best = find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
the maximum value of tpr*(1-fpr) 0.3453890206312455 for threshold 0.849
the maximum value of tpr*(1-fpr) 0.28778722018976594 for threshold 0.849
```

In [87]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_best, best_train_best)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_best, best_test_best)))
```

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

[[1227  814]
 [4833 6526]]
Test confusion matrix
[[ 513  493]
 [2437 3157]]
```

Tfidf SelectKBest Test Confusion matrix

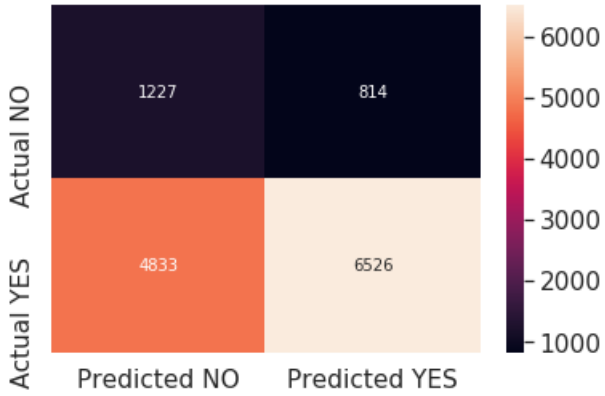
In [96]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
confusion_train=pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred_best,
best_train_best)),range(2),range(2))
confusion_train.columns = ['Predicted NO','Predicted YES']
confusion_train = confusion_train.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(confusion_train,annot=True,annot_kws={"size":10},fmt='g')
```

==========

Out[96]:

<matplotlib.axes._subplots.AxesSubplot at 0x7efc58bce198>



Tfidf SelectKbest Test Confusion matrix

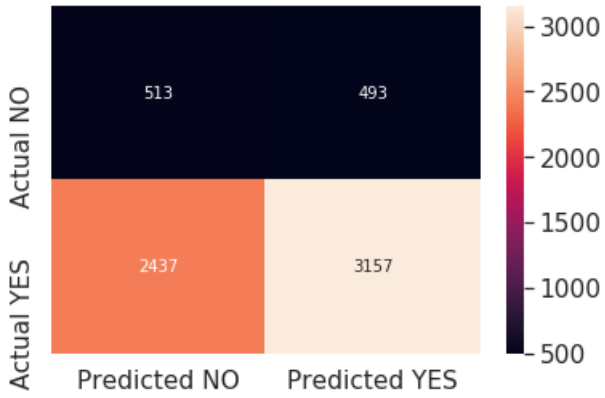
In [97]:

```
print("="*100)
confusion_test=pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred_best, best_tes
t_best)),range(2),range(2))
confusion_test.columns = ['Predicted NO','Predicted YES']
confusion_test = confusion_test.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.heatmap(confusion_test,annot=True,annot_kws={"size":10},fmt='g')
```

==========

Out[97]:

<matplotlib.axes._subplots.AxesSubplot at 0x7efc58d41ba8>



3. Conclusions

In [98]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper parameter", "Train AUC","Test AUC"]
x.add_row(["BOW", "Brute", 91,0.65,0.60]),
x.add_row(["TFIDF", "Brute", 91,0.62,0.56])
x.add_row(["Average W2V", "Brute", 91,0.64,0.59])
x.add_row(["TFIDF W2V", "Brute", 93,0.63,0.60])
```

```
print(x)
```

```
+-------------+-------+-----------------+-----------+----------+
|  Vectorizer | Model | Hyper parameter | Train AUC | Test AUC |
+-------------+-------+-----------------+-----------+----------+
|     BOW     | Brute |        91       |    0.65   |   0.6    |
|    TFIDF    | Brute |        91       |    0.62   |   0.56   |
| Average W2V | Brute |        91       |    0.64   |   0.59   |
|  TFIDF W2V  | Brute |        93       |    0.63   |   0.6    |
+-------------+-------+-----------------+-----------+----------+
```

1. The Accuracy are calculated based on the ROC Curves. 2.The values in the test Confusion matrix is almost half of the values in the train confusion matrix.

Difference between fit(),transform() and fit_transfom()

- **fit()** -It used for the learning the train data i.e,unigarams,bigrams.
- **transform()** -It used to apply the data on the learned parameters.
- **fit_transform()** -First it used for learning the data and then the data is applied on the learned parameters.

Here I used fit_transform() on the train data to learn the traindata and it applied on it for BOW..etc. And then the transform() is used apply the test data on the learned parameters.

ROC Using KNN and AUC

- Receiver Operating Curve(ROC) is a probabilistic curve which is plotted with FPR as X-axis and TPR as y-axis. Area UnderCurve(AUC) is adegree or measure of seperability.
- Higher the AUC,better the model.

Selecting the best k value for the test data using ROC and AUC as follows:

- Here I used RandomizedSearchCV with estimator as KNN and CV=3 on train data. The CV is used to divide the train data into 3 parts and each part acts as test when the other two datas are train data.
- The AUC is increases as the K value increases and decreases if K value still increases. The Optimum and best k value is chosen so that the AUC should be high.
- The Best_K value is applied on the test data and their ROC is plotted to get the AUC and to know the model is accurate or not.

SelectKBest

The top K features are selected and removing other features helps for faster computancy but accuracy is low.