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 [1]:

```
%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
```

```
C:\Users\Santosh\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows;
aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

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.

1.1 Reading Data

In [2]:

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

In [3]:

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

```
Number of data points in train data (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

```
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[4]:

	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 [5]:

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

Out[5]:

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

In [6]:

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

Out[6]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [7]:

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

In [8]:

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

Out[8]:

	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 [9]:

```
#presence of the numerical digits in a strings with numeric :
https://stackoverflow.com/a/19859308/8089731
def hasNumbers(inputString):
    return any(i.isdigit() for i in inputString)
p1 = project_data[['id','project_resource_summary']]
p1 = pd.DataFrame(data=p1)
p1.columns = ['id','digits_in_summary']
p1['digits_in_summary'] =  p1['digits_in_summary'].map(hasNumbers)
# https://stackoverflow.com/a/17383325/8089731
p1['digits_in_summary'] = p1['digits_in_summary'].astype(int)
project_data = pd.merge(project_data, p1, on='id', how='left')
project_data.head(5)
```

Out[9]:

	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

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

1.2 preprocessing of project_subject_categories

In [10]:

```
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)
project_data.head(5)
```

Out[10]:

	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
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

Unnamed:

In [11]:

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

Out[11]:

```
Counter({'Literacy_Language': 23998,
         'History_Civics': 2689,
         'Health_Sports': 6538,
         'Math_Science': 18874,
         'SpecialNeeds': 6233,
         'AppliedLearning': 5569,
         'Music_Arts': 4699,
         'Warmth': 643,
         'Care_Hunger': 643})
```

In [12]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
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 [13]:

```
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())
```

In [14]:

```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[14]:

	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

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

In [15]:

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

In [16]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 Text preprocessing

In [17]:

```
# 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 [18]:

```
# 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 [19]:

```
# 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', "do
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 [20]:

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

```
100%|████████████████████████████████████████████████████████████████████████████| 50000/50000 [04:
02<00:00, 302.06it/s]
```

In [21]:

```
preprocessed_essays[2000]
```

Out[21]:

```
'describing students not easy task many would say inspirational creative hard working they unique
unique interests learning abilities much what common desire learn day despite difficulties
encounter our classroom amazing understand everyone learns pace as teacher i pride making sure stu
dents always engaged motivated inspired create learning this project help students choose seating
appropriate developmentally many students tire sitting chairs lessons different seats available he
lps keep engaged learning flexible seating important classroom many students struggle attention fo
cus engagement we currently stability balls seating well regular chairs stools help students
trouble balance find difficult sit stability ball long period time we excited try stools part enga
ging classroom community nannan'
```

In [22]:

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

```
100%|███████████████████████████████████████████████████████████████████████████| 50000/50000
[00:13<00:00, 3672.81it/s]
```

In [23]:

```
preprocessed_titles[2000]
```

Out[23]:

```
'steady stools active learning'
```

In [24]:

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

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

project_grade_cat_list = []
for i in tqdm(project_grade_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('&','_')
    project_grade_cat_list.append(temp.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:00<00:00, 65805.60it/s]
```

In [25]:

```
project_data['clean_project_grade_category'] = project_grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head()
```

Out[25]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_title	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	I E
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Soccer Equipment for AWESOME Middle School Stu...	cl a
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Techie Kindergarteners	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Interactive Math Tools	g r

5 rows × 21 columns

In [26]:

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

Out[26]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_title	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	M
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Soccer Equipment for AWESOME Middle School Stu...	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Techie Kindergarteners	M
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Interactive Math Tools	

In [27]:

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

In [28]:

```
#Replacing Nan's with maximum occured value: https://stackoverflow.com/a/51053916/8089731
project_data['teacher_prefix'].value_counts().argmax()
project_data.fillna(value=project_data['teacher_prefix'].value_counts().argmax(),axis=1,inplace=Tru
e)
```

1.5 Preparing data for models

In [29]:

```
project_data.columns
```

Out[29]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'digits_in_summary', 'clean_categories',
       'clean_subcategories', 'essay', 'clean_project_grade_category',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

In [30]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
#from sklearn.cross_validation import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection
```

2. K Nearest Neighbor¶

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

In [31]:

```
X_train, X_test, y_train, y_test = train_test_split(project_data,project_data['project_is_approved'
], test_size=0.33, stratify = project_data['project_is_approved'])
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)

X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

Vectorizing Categorical data

one hot encodig

In [32]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True
)
vectorizer.fit(X_train['clean_categories'].values)
print(vectorizer.get_feature_names())


categories_one_hot_xtrain = vectorizer.transform(X_train['clean_categories'].values)
categories_one_hot_xcv = vectorizer.transform(X_cv['clean_categories'].values)
categories_one_hot_xtest = vectorizer.transform(X_test['clean_categories'].values)
print("Shape of matrix after one hot encodig_xtrain ",categories_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",categories_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",categories_one_hot_xtest.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig_xtrain  (22445, 9)
Shape of matrix after one hot encodig_xcv  (11055, 9)
Shape of matrix after one hot encodig_xtest  (16500, 9)
```

In [33]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=
True)
vectorizer.fit(X_train['clean_subcategories'].values)
```

```
print(vectorizer.get_feature_names())


sub_categories_one_hot_xtrain = vectorizer.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_xcv = vectorizer.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_xtest = vectorizer.transform(X_test['clean_subcategories'].values)
print("Shape of matrix after one hot encodig_xtrain ",sub_categories_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",sub_categories_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",sub_categories_one_hot_xtest.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig_xtrain  (22445, 30)
Shape of matrix after one hot encodig_xcv  (11055, 30)
Shape of matrix after one hot encodig_xtest  (16500, 30)
```

In [34]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer( lowercase=False, binary=True)
vectorizer.fit(X_train['school_state'].values)
print(vectorizer.get_feature_names())


school_state_one_hot_xtrain = vectorizer.transform(X_train['school_state'].values)
school_state_one_hot_xcv = vectorizer.transform(X_cv['school_state'].values)
school_state_one_hot_xtest = vectorizer.transform(X_test['school_state'].values)
print("Shape of matrix after one hot encodig_train ",school_state_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_cv ",school_state_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_test ",school_state_one_hot_xtest.shape)
```

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

In [35]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer( lowercase=False, binary=True)
vectorizer.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
teacher_prefix_one_hot_xtrain = vectorizer.transform(X_train['teacher_prefix'].values.astype('U'))
teacher_prefix_one_hot_xcv = vectorizer.transform(X_cv['teacher_prefix'].values.astype('U'))
teacher_prefix_one_hot_xtest = vectorizer.transform(X_test['teacher_prefix'].values.astype('U'))
print("Shape of matrix after one hot encodig_xtrain ",teacher_prefix_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",teacher_prefix_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",teacher_prefix_one_hot_xtest.shape)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig_xtrain  (22445, 5)
Shape of matrix after one hot encodig_xcv  (11055, 5)
Shape of matrix after one hot encodig_xtest  (16500, 5)
```

In [36]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
# https://stackoverflow.com/a/38161028/8089731
pattern = "(?u)\\b[\\w-]+\\b"
vectorizer = CountVectorizer(token_pattern=pattern, lowercase=False, binary=True)
```

```
vectorizer.fit(X_train['clean_project_grade_category'].values)
print(vectorizer.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
project_grade_cat_one_hot_xtrain = vectorizer.transform(X_train['clean_project_grade_category'].va
lues)
project_grade_cat_one_hot_xcv = vectorizer.transform(X_cv['clean_project_grade_category'].values)
project_grade_cat_one_hot_xtest =
vectorizer.transform(X_test['clean_project_grade_category'].values)
print("Shape of matrix after one hot encodig_xtrain ",project_grade_cat_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",project_grade_cat_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",project_grade_cat_one_hot_xtest.shape)
```

```
['Grades3-5', 'Grades6-8', 'Grades9-12', 'GradesPreK-2']
Shape of matrix after one hot encodig_xtrain  (22445, 4)
Shape of matrix after one hot encodig_xcv  (11055, 4)
Shape of matrix after one hot encodig_xtest  (16500, 4)
```

Vectorizing Numerical features

In [37]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
price_standardized_xtrain = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_standardized_xcv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_standardized_xtest = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
print("shape of price_standardized_xtrain",price_standardized_xtrain.shape)
print("shape of price_standardized_xcv",price_standardized_xcv.shape)
print("shape of price_standardized_xtest",price_standardized_xtest.shape)
```

```
shape of price_standardized_xtrain (22445, 1)
shape of price_standardized_xcv (11055, 1)
shape of price_standardized_xtest (16500, 1)
```

In [38]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

quantity_scalar = StandardScaler()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data
# print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation :
{np.sqrt(quantity_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
quantity_standardized_xtrain = quantity_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
quantity_standardized_xcv = quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
```

```
quantity_standardized_xtest = quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
print("shape of quantity_standardized_xtrain",quantity_standardized_xtrain.shape)
print("shape of quantity_standardized_xcv",quantity_standardized_xcv.shape)
print("shape of quantity_standardized_xtest",quantity_standardized_xtest.shape)
```

```
C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

```
shape of quantity_standardized_xtrain (22445, 1)
shape of quantity_standardized_xcv (11055, 1)
shape of quantity_standardized_xtest (16500, 1)
```

In [39]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

teacher_num_prev_projects_scalar = StandardScaler()
teacher_num_prev_projects_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].value
s.reshape(-1,1)) # finding the mean and standard deviation of this data
# print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_[0]}, Standard deviatio
n : {np.sqrt(teacher_number_of_previously_posted_projects_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
teacher_num_prev_projects_standardized_xtrain = teacher_num_prev_projects_scalar.transform(X_train
['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_num_prev_projects_standardized_xcv = teacher_num_prev_projects_scalar.transform(X_cv['teac
her_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_num_prev_projects_standardized_xtest = teacher_num_prev_projects_scalar.transform(X_test['
teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
print(" shape of
teacher_number_of_previously_posted_projects_standardized_xtrain",teacher_num_prev_projects_standar
ized_xtrain.shape)
print(" shape of
teacher_number_of_previously_posted_projects_standardized_xcv",teacher_num_prev_projects_standardi
d_xcv.shape)
print(" shape of
teacher_number_of_previously_posted_projects_standardized_xtest",teacher_num_prev_projects_standar
zed_xtest.shape)
```

```
C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:
```

```
Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

```
 shape of teacher_number_of_previously_posted_projects_standardized_xtrain (22445, 1)
 shape of teacher_number_of_previously_posted_projects_standardized_xcv (11055, 1)
 shape of teacher_number_of_previously_posted_projects_standardized_xtest (16500, 1)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [40]:

```
# 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
```

Vectorizing Text data

BOW on eassay

In [41]:

```
# BOW on eassay
# We are considering only the words which appeared in at least 10 documents(rows or projects).

vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(X_train['preprocessed_essays'])

essay_text_bow_xtrain = vectorizer.transform(X_train['preprocessed_essays'])
essay_text_bow_xcv = vectorizer.transform(X_cv['preprocessed_essays'])
essay_text_bow_xtest = vectorizer.transform(X_test['preprocessed_essays'])

print("Shape of matrix after BOW_text_essay X_train ",essay_text_bow_xtrain.shape)
print("Shape of matrix after BOW_text_essay X_cv ",essay_text_bow_xcv.shape)
print("Shape of matrix after BOW_text_essay X_test ",essay_text_bow_xtest.shape)
```

```
Shape of matrix after BOW_text_essay X_train  (22445, 8927)
Shape of matrix after BOW_text_essay X_cv  (11055, 8927)
Shape of matrix after BOW_text_essay X_test  (16500, 8927)
```

BOW on project_title

In [42]:

```
# BOW on project_title
# We are considering only the words which appeared in at least 10 documents(rows or projects).

vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(X_train['preprocessed_titles'])
```

```
proj_title_bow_xtrain = vectorizer.transform(X_train['preprocessed_titles'])
proj_title_bow_xcv = vectorizer.transform(X_cv['preprocessed_titles'])
proj_title_bow_xtest = vectorizer.transform(X_test['preprocessed_titles'])

print("Shape of matrix after BOW project_title_xtrain ",proj_title_bow_xtrain.shape)
print("Shape of matrix after BOW project_title_xcv ",proj_title_bow_xcv.shape)
print("Shape of matrix after BOW project_title_xtest ",proj_title_bow_xtest.shape)
```

```
Shape of matrix after BOW project_title_xtrain  (22445, 1213)
Shape of matrix after BOW project_title_xcv  (11055, 1213)
Shape of matrix after BOW project_title_xtest  (16500, 1213)
```

TFIDF Vectorizer on Essay

In [43]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['preprocessed_essays'])

essay_tfidf_xtrain = vectorizer.transform(X_train['preprocessed_essays'])
essay_tfidf_xcv = vectorizer.transform(X_cv['preprocessed_essays'])
essay_tfidf_xtest = vectorizer.transform(X_test['preprocessed_essays'])

print("Shape of matrix after tfidf eassay_xtrain ",essay_tfidf_xtrain.shape)
print("Shape of matrix after tfidf essay_xcv ",essay_tfidf_xcv.shape)
print("Shape of matrix after tfidf essay_xtest ",essay_tfidf_xtest.shape)
```

```
Shape of matrix after tfidf eassay_xtrain  (22445, 8927)
Shape of matrix after tfidf essay_xcv  (11055, 8927)
Shape of matrix after tfidf essay_xtest  (16500, 8927)
```

TFIDF Vectorizer on Project Title

In [44]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['preprocessed_titles'])

proj_title_tfidf_xtrain = vectorizer.transform(X_train['preprocessed_titles'])
proj_title_tfidf_xcv = vectorizer.transform(X_cv['preprocessed_titles'])
proj_title_tfidf_xtest = vectorizer.transform(X_test['preprocessed_titles'])

print("Shape of matrix after tfidf proj_title_xtrain ",proj_title_tfidf_xtrain.shape)
print("Shape of matrix after tfidf proj_title_xcv ",proj_title_tfidf_xcv.shape)
print("Shape of matrix after tfidf proj_title_xtest ",proj_title_tfidf_xtest.shape)
```

```
Shape of matrix after tfidf proj_title_xtrain  (22445, 1213)
Shape of matrix after tfidf proj_title_xcv  (11055, 1213)
Shape of matrix after tfidf proj_title_xtest  (16500, 1213)
```

Using Pretrained Models: Avg W2V

In [45]:

```
# 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('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

Average Word2Vec on Essay

In [46]:

```
# average Word2Vec
# compute average word2vec for each review.

# average Word2Vec on X_train
essay_avg_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    essay_avg_w2v_vectors_xtrain.append(vector)

print(len(essay_avg_w2v_vectors_xtrain))
print(len(essay_avg_w2v_vectors_xtrain[0]))

# average Word2Vec on X_cv

essay_avg_w2v_vectors_xcv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    essay_avg_w2v_vectors_xcv.append(vector)

print(len(essay_avg_w2v_vectors_xcv))
print(len(essay_avg_w2v_vectors_xcv[0]))

# average Word2Vec on X_test

essay_avg_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    essay_avg_w2v_vectors_xtest.append(vector)

print(len(essay_avg_w2v_vectors_xtest))
print(len(essay_avg_w2v_vectors_xtest[0]))
```

```
100%|████████████████████████████████████████████████████████████| 22445/22445 [00:
50<00:00, 446.43it/s]
```

```
22445
300
```

```
100%|████████████████████████████████████████████████████████████| 11055/11055 [00:
19<00:00, 565.56it/s]
```

```
11055
300
```

```
100%|████████████████████████████████████████████████████████████| 16500/16500 [00:
30<00:00, 543.11it/s]
```

```
16500
```

```
300
```

Average Word2Vec on Project Title

In [47]:

```
# average Word2Vec

# compute average word2vec for each review.

# average Word2Vec on X_train

proj_title_avg_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(X_train['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    proj_title_avg_w2v_vectors_xtrain.append(vector)

print(len(proj_title_avg_w2v_vectors_xtrain))
print(len(proj_title_avg_w2v_vectors_xtrain[0]))

# average Word2Vec on X_cv

proj_title_avg_w2v_vectors_xcv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    proj_title_avg_w2v_vectors_xcv.append(vector)

print(len(proj_title_avg_w2v_vectors_xcv))
print(len(proj_title_avg_w2v_vectors_xcv[0]))

# average Word2Vec on X_test

proj_title_avg_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(X_test['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    proj_title_avg_w2v_vectors_xtest.append(vector)

print(len(proj_title_avg_w2v_vectors_xtest))
print(len(proj_title_avg_w2v_vectors_xtest[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445
[00:01<00:00, 11273.16it/s]
```

```
22445
300
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11055/11055
[00:00<00:00, 12133.57it/s]
```

```
11055
300
```

```
100%|████████████████████████████████████████████████████████████████████████████| 16500/16500
[00:01<00:00, 8834.26it/s]
```

```
16500
300
```

Using Pretrained Models: TFIDF weighted W2V

TFIDF weighted W2V on Essays

In [48]:

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

In [49]:

```
# average Word2Vec

# average Word2Vec on X_train

essay_tfidf_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors_xtrain.append(vector)

print(len(essay_tfidf_w2v_vectors_xtrain))
print(len(essay_tfidf_w2v_vectors_xtrain[0]))

# average Word2Vec on X_cv
essay_tfidf_w2v_vectors_xcv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors_xcv.append(vector)

print(len(essay_tfidf_w2v_vectors_xcv))
print(len(essay_tfidf_w2v_vectors_xcv[0]))
```

```
# average Word2Vec on X_train
essay_tfidf_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors_xtest.append(vector)

print(len(essay_tfidf_w2v_vectors_xtest))
print(len(essay_tfidf_w2v_vectors_xtest[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 22445/22445 [05
:22<00:00, 69.69it/s]
```

```
22445
300
```

```
100%|████████████████████████████████████████████████████████████████████████| 11055/11055 [03
:22<00:00, 54.69it/s]
```

```
11055
300
```

```
100%|████████████████████████████████████████████████████████████████████████| 16500/16500 [03
:15<00:00, 84.49it/s]
```

```
16500
300
```

TFIDF weighted W2V on Project Title

In [50]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
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_words = set(tfidf_model.get_feature_names())
```

In [51]:

```
# TFIDF weighted W2V on Project Title
# compute average word2vec for each review.

# TFIDF weighted W2V on X_train

proj_title_tfidf_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(X_train['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
```

```
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    proj_title_tfidf_w2v_vectors_xtrain.append(vector)

print(len(proj_title_tfidf_w2v_vectors_xtrain))
print(len(proj_title_tfidf_w2v_vectors_xtrain[0]))


# TFIDF weighted W2V on X_cv
proj_title_tfidf_w2v_vectors_xcv = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(X_cv['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    proj_title_tfidf_w2v_vectors_xcv.append(vector)

print(len(proj_title_tfidf_w2v_vectors_xcv))
print(len(proj_title_tfidf_w2v_vectors_xcv[0]))


# TFIDF weighted W2V on X_test
proj_title_tfidf_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(X_test['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    proj_title_tfidf_w2v_vectors_xtest.append(vector)

print(len(proj_title_tfidf_w2v_vectors_xtest))
print(len(proj_title_tfidf_w2v_vectors_xtest[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████| 22445/22445
[00:03<00:00, 5733.87it/s]
```

```
22445
300
```

```
100%|████████████████████████████████████████████████████████████████████████████| 11055/11055
[00:01<00:00, 5775.03it/s]
```

```
11055
300
```

```
100%|████████████████████████████████████████████████████████████████████████████| 16500/16500
[00:02<00:00, 6067.49it/s]
```

```
16500
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 [52]:

```
# 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 [113]:

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

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


X_train1=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
school_state_one_hot_xtrain,
                 teacher_prefix_one_hot_xtrain, project_grade_cat_one_hot_xtrain,
price_standardized_xtrain,
               teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                essay_text_bow_xtrain, proj_title_bow_xtrain)).tocsr()


X_cv1=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
                school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
                project_grade_cat_one_hot_xcv, price_standardized_xcv,
               teacher_num_prev_projects_standardized_xcv, quantity_standardized_xcv,
                essay_text_bow_xcv, proj_title_bow_xcv)).tocsr()


X_test1=hstack((categories_one_hot_xtest, sub_categories_one_hot_xtest,
                school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
                project_grade_cat_one_hot_xtest, price_standardized_xtest,
               teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                essay_text_bow_xtest, proj_title_bow_xtest)).tocsr()

print(X_train1.shape, y_train.shape)
print(X_cv1.shape, y_cv.shape)
print(X_test1.shape, y_test.shape)
```

```
(22445, 10242) (22445,)
(11055, 10242) (11055,)
(16500, 10242) (16500,)
```

Simple for loop (if you are having memory limitations use this)

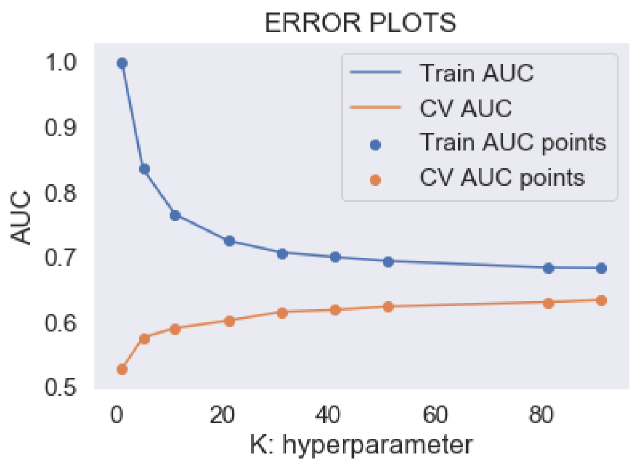
In [114]:

```
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 cr_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
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [115]:

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

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


train_auc = []
cv_auc = []
K = [1, 5, 11, 21, 31, 41, 51, 81, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i,n_jobs=-1)
    neigh.fit(X_train1[:,:],y_train[:])

    y_train_pred = batch_predict(neigh, X_train1[:,:])
    y_cv_pred = batch_predict(neigh, X_cv1[:,:])

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

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

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

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

```
100%|██████████████████████████████████████████████████████████| 9/9 [36:
51<00:00, 228.82s/it]
```



Conclusion:¶

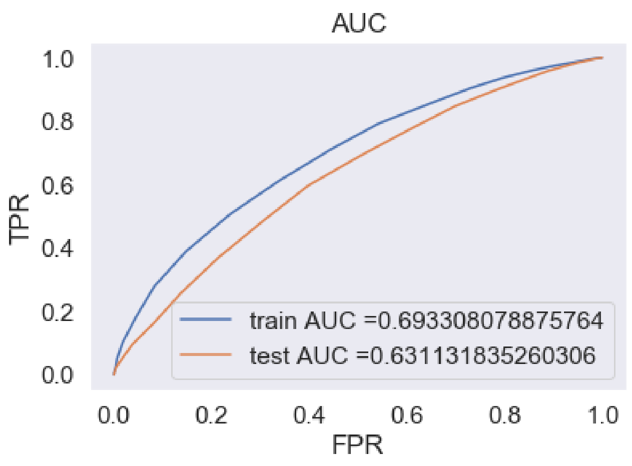
For CV Max value of AUC is around 0.64 for K=51, so considering best K=51 for Test Data

In [116]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Select best K
best_k_set_bow = 51
```

In [118]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k_set_bow, n_jobs=-1)
neigh.fit(X_train1[:,:], y_train[:])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train1[:,:])
y_test_pred = batch_predict(neigh, X_test1[:])

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

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



Conclusion

```
1. For both Train and Test Data FPR and TPR is Maximum at 1
2. For both Train and Test Data FPR and TPR is Minimum at 0
3. We need greater TPR and Lesser FPR value, for that we need thrushold values
```

In [119]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, 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))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [120]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24823468990989186 for threshold 0.784
[[ 1586  1877]
 [ 3950 15032]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24993196666026957 for threshold 0.804
[[1252 1294]
 [4277 9677]]
```

Confusion matrix for train data

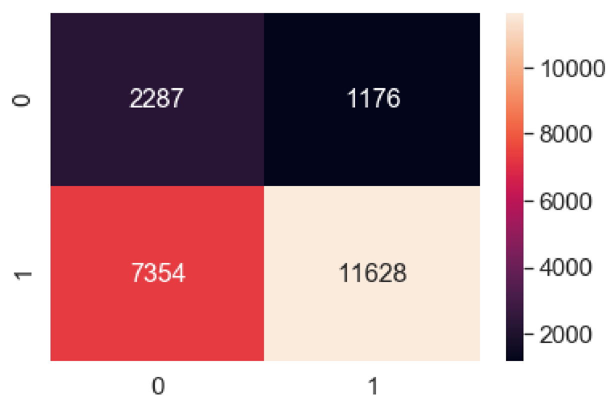
In [121]:

```
# Confusion matrix for train data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtrain =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred,
tr_thresholds, train_fpr, train_tpr)))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtrain, annot=True,annot_kws={"size": 16}, fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.40455421192829977 for threshold 0.824
```

Out[121]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548d4cc898>
```



Conclusion

1. True Positive Rate is High as well as False Positive Rate is also high whih is not desirable
2. so Using Bag of Words We have both TPR and FPR high

Confusion matrix for test data

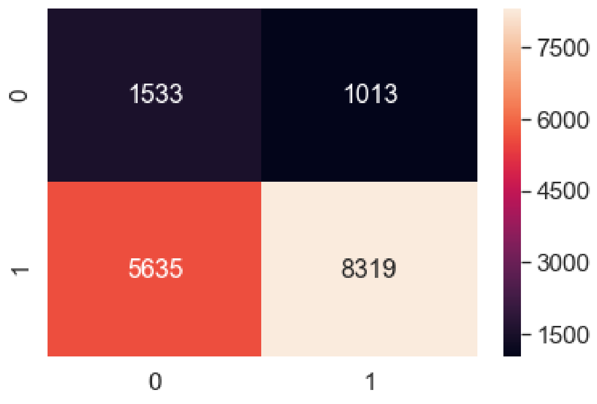
In [122]:

```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtest, annot=True,annot_kws={"size": 16}, fmt='g')#font size
```

```
the maximum value of tpr*(1-fpr) 0.35896835196692173 for threshold 0.824
```

Out[122]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548c158160>
```



Conclusion

```
1.For Test Data using Bag of words vectorization Both TPR and FPR is HIgh
2.The result is not Desirable
```

2.4.2 Applying KNN brute force on TFIDF, SET 2

In [123]:

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

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

X_train2=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
                school_state_one_hot_xtrain, teacher_prefix_one_hot_xtrain,
                project_grade_cat_one_hot_xtrain, price_standardized_xtrain,
                 teacher_num_prev_projects_standardized_xtrain,
                 quantity_standardized_xtrain,essay_tfidf_xtrain, proj_title_tfidf_xtrain)).tocsr()


X_cv2=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
                school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
                project_grade_cat_one_hot_xcv, price_standardized_xcv,
               teacher_num_prev_projects_standardized_xcv,quantity_standardized_xcv,
                essay_tfidf_xcv, proj_title_tfidf_xcv)).tocsr()


X_test2=hstack((categories_one_hot_xtest, sub_categories_one_hot_xtest,
                school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
                project_grade_cat_one_hot_xtest, price_standardized_xtest,
               teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                essay_tfidf_xtest, proj_title_tfidf_xtest)).tocsr()
```

```
print(X_train2.shape)
print(X_cv2.shape)
print(X_test2.shape)
```

```
(22445, 10242)
(11055, 10242)
(16500, 10242)
```

In [124]:

```
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 cr_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
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

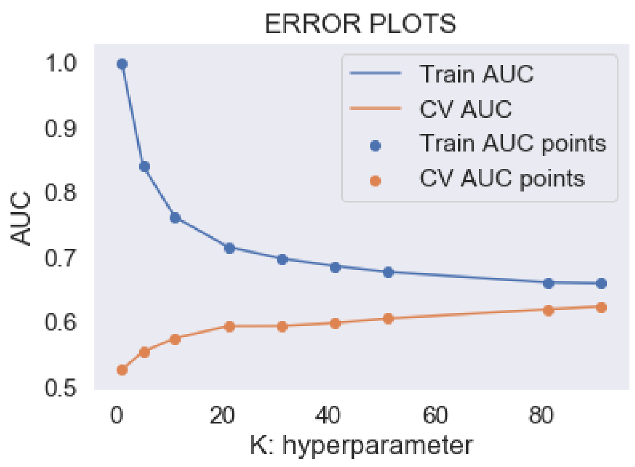
Appling KNN SET 2: TFIDF featurization

K: Hyper parameter Tuning Set 2

In [125]:

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


train_auc = []
cv_auc = []

K = [1, 5, 11, 21, 31, 41, 51, 81, 91]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_train2[:,:],y_train[:])

    y_train_pred = batch_predict(neigh, X_train2[:,:])
    y_cv_pred = batch_predict(neigh, X_cv2[:,:])

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

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

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

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

```
100%|██████████████████████████████████████████████████████████████████████| 9/9 [41:
33<00:00, 279.78s/it]
```



Conclusion

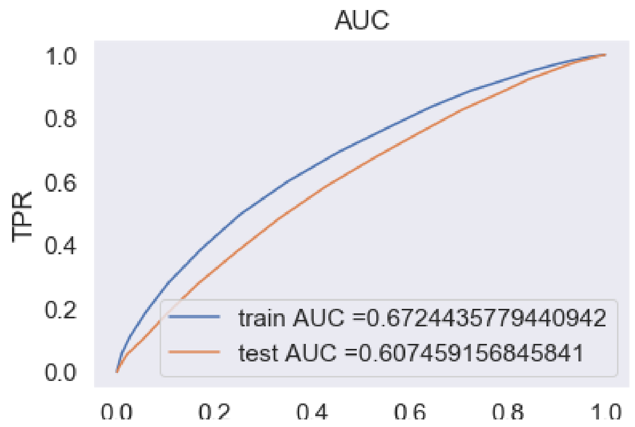
Considering K= 61 as best parameter for set 2 to test the data

In [126]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Selecting best K
best_k_set_tfidf = 61
```

In [127]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k_set_tfidf, n_jobs =-1)
neigh.fit(X_train2[:,:], y_train[:])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train2[:,:])
y_test_pred = batch_predict(neigh, X_test2[:])

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

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



FPR

In [128]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, 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))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [129]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.3903655644953 for threshold 0.852
[[ 2246  1217]
 [ 7557 11425]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3344124973076727 for threshold 0.852
[[1454 1092]
 [5783 8171]]
```

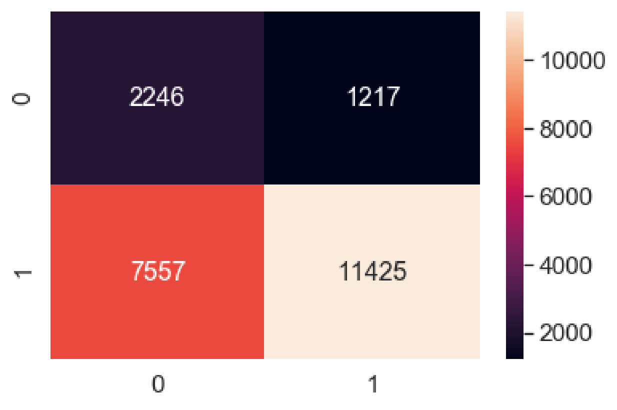
In [130]:

```
# Confusion Matrix for Train Data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtrain =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred,
tr_thresholds, train_fpr, train_tpr)))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtrain, annot=True,annot_kws={"size": 16}, fmt='g')#font size
```

```
the maximum value of tpr*(1-fpr) 0.3903655644953 for threshold 0.852
```

Out[130]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548cb0d710>
```



Conclusion

```
1. For Set 2 tfidf vectorization TPR is incresed and FPR is reduced compared to set 1.
2. Still FPR is High and it is not desirable result
```

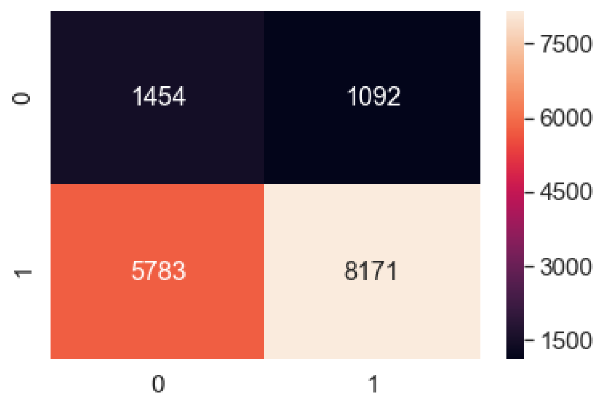
In [131]:

```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtest, annot=True,annot_kws={"size": 16}, fmt='g')#font size
```

```
the maximum value of tpr*(1-fpr) 0.3344124973076727 for threshold 0.852
```

Out[131]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548caa9438>
```



2.4.3 Applying KNN brute force on AVG W2V, SET 3

In [81]:

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

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


X_train3=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
                school_state_one_hot_xtrain, teacher_prefix_one_hot_xtrain,
                project_grade_cat_one_hot_xtrain, price_standardized_xtrain,
               teacher_num_prev_projects_standardized_xtrain,
                essay_avg_w2v_vectors_xtrain, proj_title_avg_w2v_vectors_xtrain)).tocsr()


X_cv3=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
                school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
                project_grade_cat_one_hot_xcv, price_standardized_xcv,
               teacher_num_prev_projects_standardized_xcv,
                essay_avg_w2v_vectors_xcv, proj_title_avg_w2v_vectors_xcv)).tocsr()


X_test3=hstack((categories_one_hot_xtest, sub_categories_one_hot_xtest,
                school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
                project_grade_cat_one_hot_xtest, price_standardized_xtest,
               teacher_num_prev_projects_standardized_xtest,
                essay_avg_w2v_vectors_xtest, proj_title_avg_w2v_vectors_xtest)).tocsr()
```

```
print(X_train3.shape, y_train.shape)
print(X_cv3.shape, y_cv.shape)
print(X_test3.shape, y_test.shape)
```

```
(22445, 701) (22445,)
(11055, 701) (11055,)
(16500, 701) (16500,)
```

In [82]:

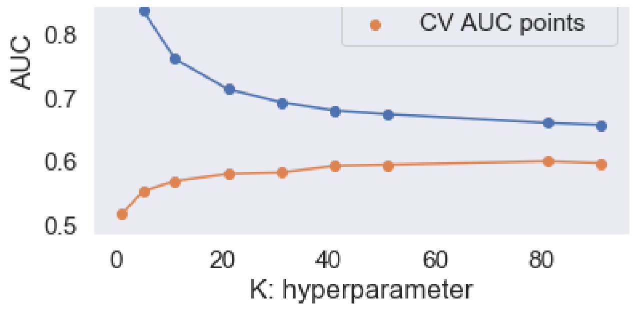
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_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
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [83]:

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


train_auc = []
cv_auc = []
K = [1, 5, 11, 21, 31, 41, 51, 81, 91]


for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs =-1)
    neigh.fit(X_train3[:5449,:],y_train[:5449])

    y_train_pred = batch_predict(neigh, X_train3[:5449,:])
    y_cv_pred = batch_predict(neigh, X_cv3[:5449,:])

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train[:5449],y_train_pred))
    cv_auc.append(roc_auc_score(y_cv[:5449], y_cv_pred))

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

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

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

```
100%|██████████████████████████████████████████████████████████████████████████| 9/9 [24:
31<00:00, 166.90s/it]
```

ERROR PLOTS
1.0
0.9
Train AUC
CV AUC
Train AUC points

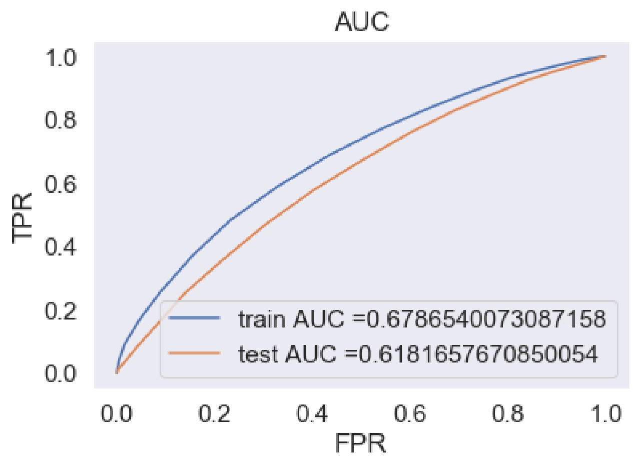


In [84]:

```
## Testing the performance of the model on test data, plotting ROC Curves
# Selecting best K
best_k_set3_AVGW2V = 57
```

In [85]:

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


neigh = KNeighborsClassifier(n_neighbors=best_k_set3_AVGW2V, n_jobs =-1)
neigh.fit(X_train3[:,:], y_train[:])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train3[:,:])
y_test_pred = batch_predict(neigh, X_test3[:])

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

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



In [87]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, 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))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [88]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.394652495838345 for threshold 0.86
[[ 2325  1138]
 [ 7824 11158]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.34476392582023235 for threshold 0.86
[[1524 1022]
 [5917 8037]]
```

In [ ]:

```
# Confusion Matrix for Train Data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtrain =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred,
tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtrain, annot=True,annot_kws={"size": 16}, fmt='g')
```

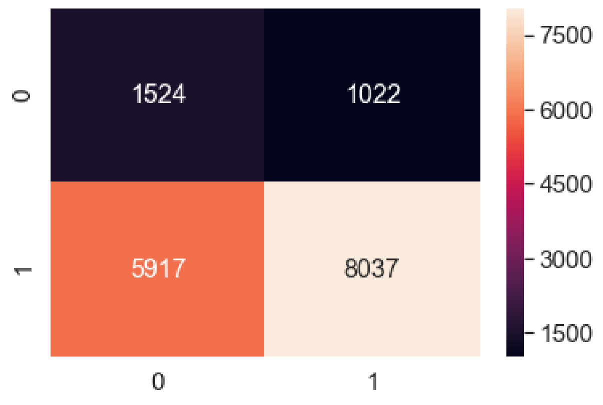
In [89]:

```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtest, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.34476392582023235 for threshold 0.86
```

Out[89]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548bf004a8>
```



2.4.4 Applying KNN brute force on TFIDF W2V, SET 4

In [90]:

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

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


X_train4=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
                school_state_one_hot_xtrain, teacher_prefix_one_hot_xtrain,
                project_grade_cat_one_hot_xtrain, price_standardized_xtrain,
               teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                essay_tfidf_w2v_vectors_xtrain, proj_title_tfidf_w2v_vectors_xtrain)).tocsr()


X_cv4=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
                school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
                project_grade_cat_one_hot_xcv, price_standardized_xcv,
               teacher_num_prev_projects_standardized_xcv, quantity_standardized_xcv,
                essay_tfidf_w2v_vectors_xcv, proj_title_tfidf_w2v_vectors_xcv)).tocsr()


X_test4=hstack((categories_one_hot_xtest, sub_categories_one_hot_xtest,
                school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
                project_grade_cat_one_hot_xtest, price_standardized_xtest,
               teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                essay_tfidf_w2v_vectors_xtest, proj_title_tfidf_w2v_vectors_xtest)).tocsr()

print(X_train4.shape, y_train.shape)
print(X_cv4.shape, y_cv.shape)
print(X_test4.shape, y_test.shape)
```

```
(22445, 702) (22445,)
(11055, 702) (11055,)
(16500, 702) (16500,)
```

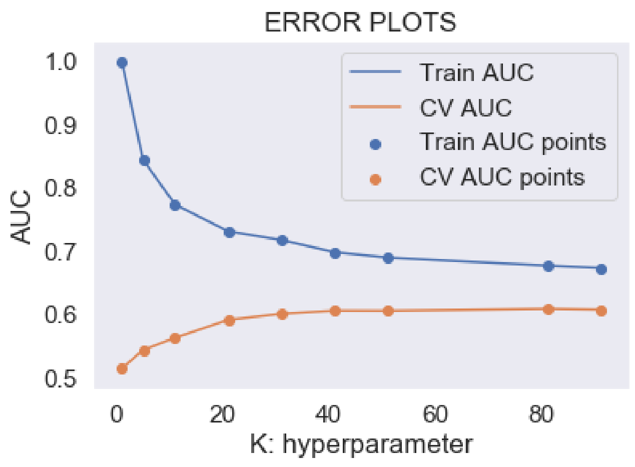
In [91]:

```
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 cr_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
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [92]:

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


train_auc = []
cv_auc = []
K = [1, 5, 11, 21, 31, 41, 51, 81, 91]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_train4[:5449,:],y_train[:5449])

    y_train_pred = batch_predict(neigh, X_train4[:5449,:])
    y_cv_pred = batch_predict(neigh, X_cv4[:5449,:])

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train[:5449],y_train_pred))
    cv_auc.append(roc_auc_score(y_cv[:5449], y_cv_pred))

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

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

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

```
100%|████████████████████████████████████████████████████████████████████| 9/9 [21:
47<00:00, 145.24s/it]
```

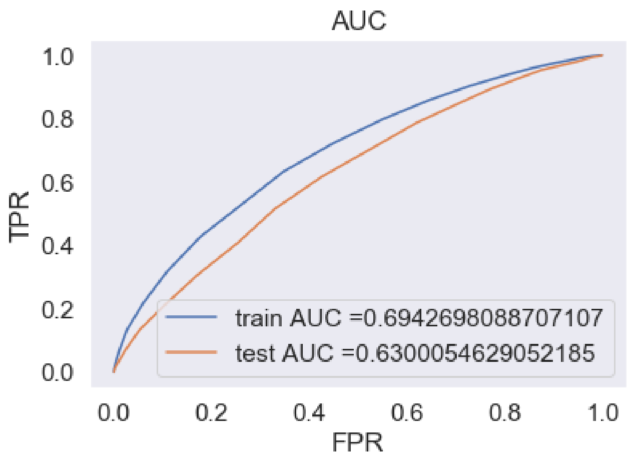


In [93]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Selecting best K
best_k_set4_TFIDFW2V = 61
```

In [94]:

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

neigh = KNeighborsClassifier(n_neighbors=best_k_set4_TFIDFW2V, n_jobs=-1)
neigh.fit(X_train4[:,:], y_train[:])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train4[:,:])
y_test_pred = batch_predict(neigh, X_test4[:])

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

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



In [95]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, 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))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [96]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4127865500982389 for threshold 0.852
[[ 2258  1205]
 [ 6965 12017]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3539016255971112 for threshold 0.852
[[1463 1083]
 [5360 8594]]
```

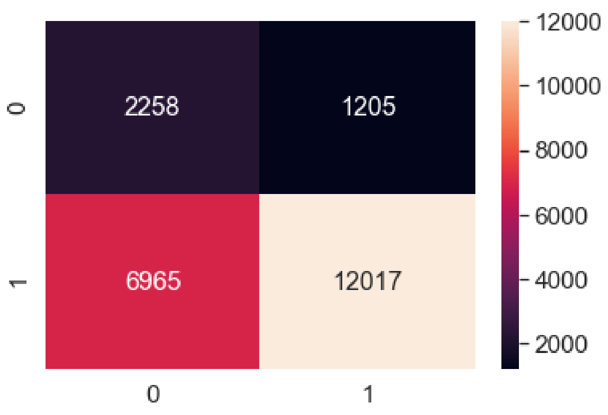
In [97]:

```
# Confusion Matrix for Train Data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtrain =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred,
tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtrain, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4127865500982389 for threshold 0.852
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2549649cfd0>
```



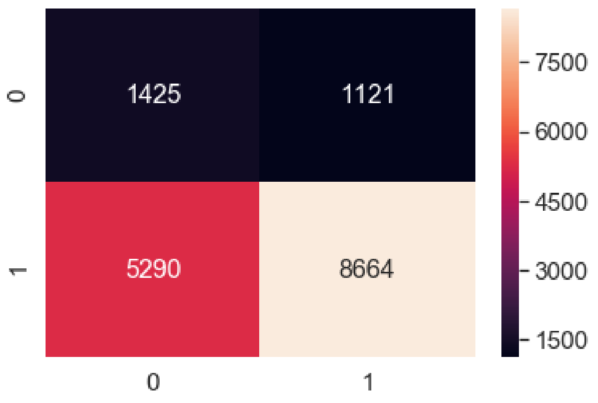
In [117]:

```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix
conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtest, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.347517108452292 for threshold 0.85
```

Out[117]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x25fbe8eae80>
```



2.5 Feature selection with `SelectKBest`

In [99]:

```
# 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 [100]:

```
# Set 2 Dimensions
print(X_train2.shape, y_train.shape)
print(X_cv2.shape, y_cv.shape)
print(X_test2.shape, y_test.shape)
```

```
(22445, 10242) (22445,)
(11055, 10242) (11055,)
(16500, 10242) (16500,)
```

In [101]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039


from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_xtrain,sub_categories_one_hot_xtrain,school_state_one_hot_xtrain
,teacher_prefix_one_hot_xtrain

,project_grade_cat_one_hot_xtrain,price_standardized_xtrain,quantity_standardized_xtrain

,teacher_num_prev_projects_standardized_xtrain,essay_tfidf_xtrain,proj_title_tfidf_xtrain)).tocsr(
).toarray()

X_cr =
hstack((categories_one_hot_xcv,sub_categories_one_hot_xcv,school_state_one_hot_xcv,teacher_prefix_o
ne_hot_xcv
               ,project_grade_cat_one_hot_xcv,price_standardized_xcv,quantity_standardized_xcv
               ,teacher_num_prev_projects_standardized_xcv,essay_tfidf_xcv,proj_title_tfidf_xcv)).t
ocsr().toarray()

X_te = hstack((categories_one_hot_xtest,sub_categories_one_hot_xtest,school_state_one_hot_xtest,te
acher_prefix_one_hot_xtest

,project_grade_cat_one_hot_xtest,price_standardized_xtest,quantity_standardized_xtest

,teacher_num_prev_projects_standardized_xtest,essay_tfidf_xtest,proj_title_tfidf_xtest)).tocsr().t
oarray()
```

In [102]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_tr = scaler.fit_transform(X_tr,y_train)
X_cr = scaler.transform(X_cr)
X_te = scaler.transform(X_te)


from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2,k=2000).fit(X_tr, y_train)
X_tr = t.transform(X_tr)
X_te = t.transform(X_te)
X_cr = t.transform(X_cr)


print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
```

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

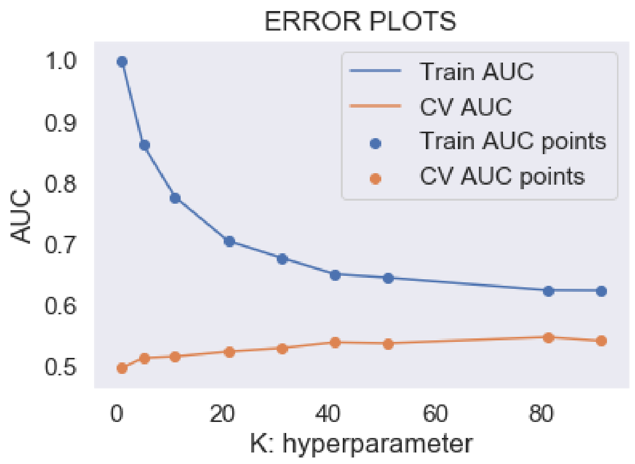
In [103]:

```
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 cr_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
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [104]:

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


train_auc = []
cv_auc = []
K = [1, 5, 11, 21, 31, 41, 51, 81, 91]

for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i)
    neigh.fit(X_tr[:5449,:],y_train[:5449])

    y_train_pred = batch_predict(neigh, X_tr[:5449,:])
    y_cv_pred = batch_predict(neigh, X_cr[:5449,:])

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train[:5449],y_train_pred))
    cv_auc.append(roc_auc_score(y_cv[:5449], y_cv_pred))

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

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

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

```
100%|██████████████████████████████████████████████████████████████████| 9/9 [28:
17<00:00, 193.41s/it]
```

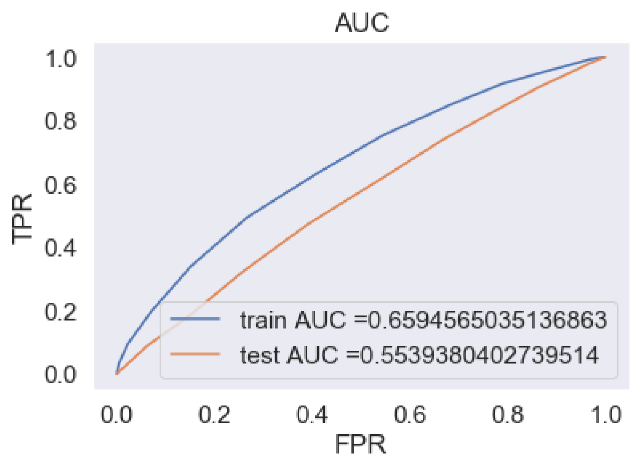


In [105]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Selecting best K
best_k_set2_tfidf_selectKbest = 55
```

In [107]:

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

neigh = KNeighborsClassifier(n_neighbors=best_k_set2_tfidf_selectKbest, n_jobs=-1)
neigh.fit(X_tr[:,:], y_train[:])
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr[:,:])
y_test_pred = batch_predict(neigh, X_te[:])

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

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



In [108]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, 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))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [110]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.3735319504019386 for threshold 0.873
[[ 2037  1426]
 [ 6928 12054]]
Test confusion matrix
```

```
the maximum value of tpr*(1-fpr) 0.2881338819357194 for threshold 0.891
[[1547  999]
 [7337 6617]]
```

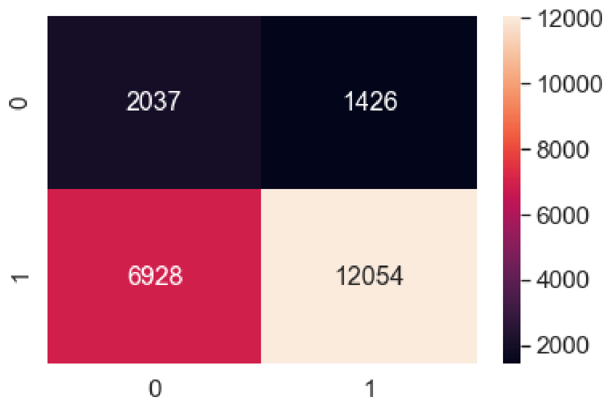
In [111]:

```
# Confusion Matrix for Train Data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtrain =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred,
tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtrain, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.3735319504019386 for threshold 0.873
```

Out[111]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548bf4bbe0>
```



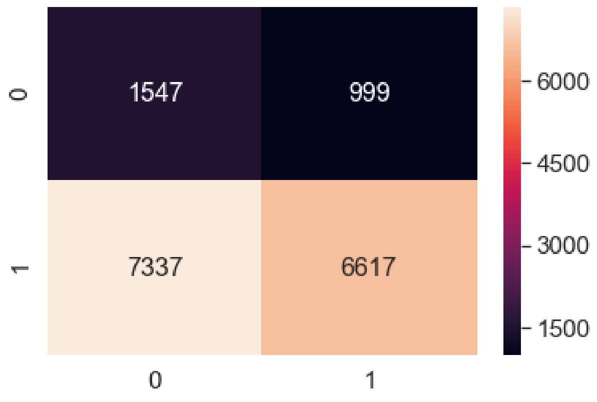
In [112]:

```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtest, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2881338819357194 for threshold 0.891
```

Out[112]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2548bf40438>
```



3. Conclusions

In [36]:

```
# Please compare all your models using Prettytable library
```

In [132]:

```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable

x = PrettyTable()

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

x.add_row(["BOW", "Brute", 51, 0.63])
x.add_row(["TFIDF", "Brute", 61, 0.60])
x.add_row(["AVG W2V", "Brute", 57, 0.61])
x.add_row(["TFIDF W2V", "Brute", 61, 0.63])

print(x)
```

```
+------------+-------+-------------------+------+
| Vectorizer | Model | K:Hyper Parameter | AUC  |
+------------+-------+-------------------+------+
|    BOW     | Brute |         51        | 0.63 |
|   TFIDF    | Brute |         61        | 0.6  |
|  AVG W2V   | Brute |         57        | 0.61 |
| TFIDF W2V  | Brute |         61        | 0.63 |
+------------+-------+-------------------+------+
```

In [ ]:

```

```