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*
`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 chart_studio.plotly import plot, iplot
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1.1 Reading Data

In [2]:

```
#Due to computational constraints,I'm only considering 20k ro
ws
project_data = pd.read_csv('train_data.csv',nrows=20000)
resource_data = pd.read_csv('resources.csv')
```

In [3]:

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

```
Number of data points in train data (20000, 17
)
--------------------------------------------------
----
The attributes of data : ['Unnamed: 0' 'id' 't
eacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_c
ategory'
 'project_subject_categories' 'project_subject
_subcategories'
 'project_title' 'project_essay_1' 'project_es
say_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects
' 'project_is_approved']
```

In [4]:

```
project_data.columns
```

Out[4]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teac
her_prefix', 'school_state',
       'project_submitted_datetime', 'project_
grade_category',
       'project_subject_categories', 'project_
subject_subcategories',
       'project_title', 'project_essay_1', 'pr
oject_essay_2',
       'project_essay_3', 'project_essay_4', '
project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved'],
      dtype='object')
```

In [5]:

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


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


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

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

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	scho
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	
7176	79341	p091436	bb2599c4a114d211b3381abe9f899bf8	Mrs.	

In [6]:

```
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[6]:

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

1.2 preprocessing of `project_subject_categories`

In [7]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-pyth
on/
# https://stackoverflow.com/questions/23669024/how-to-strip-a
-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whit
espace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth
, Care & Hunger"
    for j in i.split(','): # it will split it in three parts
["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            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 ' '(s
pace) 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, inp
lace=True)

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

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

1.3 preprocessing of `project_subject_subcategories`

In [8]:

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

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

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

```
    sub_cat_list.append(temp.strip())

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

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

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

1.4 preprocessing of `teacher_prefix`

In [9]:

```
#NaN values in techer prefix will create a problem while enco
ding,so we replace NaN values with the mode of that particula
r column
#removing dot(.) since it is a special character
mode_of_teacher_prefix = project_data['teacher_prefix'].value
_counts().index[0]


project_data['teacher_prefix'] = project_data['teacher_prefix
'].fillna(mode_of_teacher_prefix)
```

In [10]:

```
prefixes = []

for i in range(len(project_data)):
    a = project_data["teacher_prefix"][i].replace(".", "")
    prefixes.append(a)
```

In [11]:

```
project_data.drop(['teacher_prefix'], axis = 1, inplace = Tru
e)
project_data["teacher_prefix"] = prefixes
print("After removing the special characters ,Column values:
 ")
np.unique(project_data["teacher_prefix"].values)
```

```
After removing the special characters ,Column
values:
```

Out[11]:

```
array(['Mr', 'Mrs', 'Ms', 'Teacher'], dtype=ob
ject)
```

1.5 preprocessing of `school_state`

In [12]:

```
school_states = list(project_data['school_state'].values)
# remove special characters from list of strings python: http
s://stackoverflow.com/a/47301924/4084039

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

school_states_list = []
for i in school_states:
    temp = ""
    # consider we have text like this "Math & Science, Warmth
, Care & Hunger"
    for j in i.split(','): # it will split it in three parts
["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            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 ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc
", remove the trailing spaces
        temp = temp.replace('&','_')
    school_states_list.append(temp.strip())
```

```
project_data.drop(['school_state'], axis=1, inplace=True)
project_data['school_state'] = school_states_list


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

school_states_dict = dict(my_counter)
sorted_school_states_dict = dict(sorted(school_states_dict.it
ems(), key=lambda kv: kv[1]))
```

1.6 preprocessing of `project_grade_category`

In [13]:

```
# We need to get rid of The spaces between the text and the h
yphens because they're special characters.
#Rmoving multiple characters from a string in Python
#https://stackoverflow.com/questions/3411771/multiple-charact
er-replace-with-python

project_grade_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" "
, "_").replace("-", "_")
    project_grade_category.append(a)
```

In [14]:

```
project_data.drop(['project_grade_category'], axis = 1, inpla
ce = True)
project_data["project_grade_category"] = project_grade_catego
ry
print("After removing the special characters ,Column values:
 ")
np.unique(project_data["project_grade_category"].values)
```

```
After removing the special characters ,Column
values:
```

Out[14]:

```
array(['Grades_3_5', 'Grades_6_8', 'Grades_9_1
2', 'Grades_PreK_2'], dtype=object)
```

1.3 Text preprocessing

In [15]:

```
# 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 [16]:

```
len(project_data["essay"])
```

Out[16]:

```
20000
```

In [17]:

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

Out[17]:

	Unnamed: 0	id	teacher
473	100660	p234804	cbc0e38f522143b86d372f8b43d4
7176			

79341	p091436	bb2599c4a114d211b3381abe9f899bf8	2016-04-27 07:24:47	Robots ar Taking ove 2nd Grad

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]:

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

```
Wilson Elementary School is a growing, dynamic
 neighborhood school of about 350 students in
Corvallis, Oregon.  Wilson is a Title 1 school
 and has a wide range of students from diverse
 backgrounds: socioeconomically, racially, and
```

```
 culturally.  Wilson seeks to provide a rich,
engaging curriculum while encouraging positive
 behavior.  Wilson Elementary celebrates abili
ties and believes in the possibilities of stud
ents.  My classroom will be an extension of th
ese ideals and goals...a place to be brave, be
 kind, celebrate mistakes and work hard.Who kn
ew a carpet could be a key element to learning
, growth, discussion and building community?!
 Indeed that 9x12 space can be a sacred space.
  A place where real conversations can happen.
  A spot for kids to go deeper with their thin
king and wonderings.  Time on the carpet can o
ffer students a place to learn the value of ac
tive listening to truly understand another.  A
 place to practice student structured talk in
a safe environment where making mistakes is ok
 and metacognition is celebrated.  Providing a
dequate space that is clearly organized with s
et expectations can lead to deeper learning an
d opportunities for classroom bonding...like M
AGIC.nannan
==============================================
====
```

In [20]:

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

```
Wilson Elementary School is a growing, dynamic
 neighborhood school of about 350 students in
Corvallis, Oregon.  Wilson is a Title 1 school
```

```
 and has a wide range of students from diverse
 backgrounds: socioeconomically, racially, and
 culturally.  Wilson seeks to provide a rich,
engaging curriculum while encouraging positive
 behavior.  Wilson Elementary celebrates abili
ties and believes in the possibilities of stud
ents.  My classroom will be an extension of th
ese ideals and goals...a place to be brave, be
 kind, celebrate mistakes and work hard.Who kn
ew a carpet could be a key element to learning
, growth, discussion and building community?!
 Indeed that 9x12 space can be a sacred space.
  A place where real conversations can happen.
  A spot for kids to go deeper with their thin
king and wonderings.  Time on the carpet can o
ffer students a place to learn the value of ac
tive listening to truly understand another.  A
 place to practice student structured talk in
a safe environment where making mistakes is ok
 and metacognition is celebrated.  Providing a
dequate space that is clearly organized with s
et expectations can lead to deeper learning an
d opportunities for classroom bonding...like M
AGIC.
```

In [21]:

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

```
Wilson Elementary School is a growing dynamic
neighborhood school of about 350 students in C
orvallis Oregon Wilson is a Title 1 school and
 has a wide range of students from diverse bac
kgrounds socioeconomically racially and cultur
ally Wilson seeks to provide a rich engaging c
```

```
urriculum while encouraging positive behavior
Wilson Elementary celebrates abilities and bel
ieves in the possibilities of students My clas
sroom will be an extension of these ideals and
 goals a place to be brave be kind celebrate m
istakes and work hard Who knew a carpet could
be a key element to learning growth discussion
 and building community Indeed that 9x12 space
 can be a sacred space A place where real conv
ersations can happen A spot for kids to go dee
per with their thinking and wonderings Time on
 the carpet can offer students a place to lear
n the value of active listening to truly under
stand another A place to practice student stru
ctured talk in a safe environment where making
 mistakes is ok and metacognition is celebrate
d Providing adequate space that is clearly org
anized with set expectations can lead to deepe
r learning and opportunities for classroom bon
ding like MAGIC
```

In [22]:

```
# 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', 'b
etween', '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', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', '
same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't",
'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "co
uldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", '
isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't",
'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
 \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [23]:

```
# Combining all the above statements
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 stopw
ords)
```

```
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 20000/20000 [00:08<00:00, 239
9.54it/s]
```

In [24]:

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

Out[24]:

```
'wilson elementary school growing dynamic neig
hborhood school 350 students corvallis oregon
wilson title 1 school wide range students dive
rse backgrounds socioeconomically racially cul
turally wilson seeks provide rich engaging cur
riculum encouraging positive behavior wilson e
lementary celebrates abilities believes possib
ilities students my classroom extension ideals
 goals place brave kind celebrate mistakes wor
k hard who knew carpet could key element learn
ing growth discussion building community indee
d 9x12 space sacred space a place real convers
ations happen a spot kids go deeper thinking w
onderings time carpet offer students place lea
rn value active listening truly understand ano
ther a place practice student structured talk
safe environment making mistakes ok metacognit
ion celebrated providing adequate space clearl
y organized set expectations lead deeper learn
ing opportunities classroom bonding like magic
 nannan'
```

In [25]:

```
#creating a new column with the preprocessed essays and repla
cing it with the original columns
project_data['preprocessed_essays'] = preprocessed_essays
```

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

1.4 Preprocessing of $project_tit \leq$

In [26]:

```
# similarly you can preprocess the titles also
preprocessed_titles = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].values):
    sent = decontracted(sentence)
    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 stopw
ords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 20000/20000 [00:00<00:00, 549
01.60it/s]
```

In [27]:

```
#creating a new column with the preprocessed titles,useful fo
r analysis
project_data['preprocessed_titles'] = preprocessed_titles
```

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

In [28]:

```
from sklearn.model_selection import train_test_split
#How to split whole dataset into Train,CV and test
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split
project_data_train, project_data_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'])
project_data_train, project_data_cv, y_train, y_cv = train_test_split(project_data_train, y_train, test_size=0.33, stratify=y_train)
```

In [29]:

```
print("Split ratio")
print('-'*50)
print('Train dataset:',len(project_data_train)/len(project_data)*100,'%\n','size:',len(project_data_train))
print('Cross validation dataset:',len(project_data_cv)/len(project_data)*100,'%\n','size:',len(project_data_cv))
print('Test dataset:',len(project_data_test)/len(project_data)*100,'%\n','size:',len(project_data_test))
```

```
Split ratio
--------------------------------------------------
----
Train dataset: 44.89 %
 size: 8978
```

```
Cross validation dataset: 22.11 %
 size: 4422
Test dataset: 33.0 %
 size: 6600
```

In [30]:

```
#Features
project_data_train.drop(['project_is_approved'], axis=1, inpl
ace=True)
project_data_cv.drop(['project_is_approved'], axis=1, inplace
=True)
project_data_test.drop(['project_is_approved'], axis=1, inpla
ce=True)
```

1.5 Preparing data for models

In [31]:

```
project_data.columns
```

Out[31]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'Date
', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'clean_categories', 'clean_subcategorie
s', 'teacher_prefix',
       'school_state', 'project_grade_category
', 'essay',
       'preprocessed_essays', 'preprocessed_ti
tles'],
      dtype='object')
```

we are going to consider

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

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

      - quantity : numerical (optional)
      - teacher_number_of_previously_posted_projects : nu
   merical
      - price : numerical
```

Encoding numerical, categorical features

1.5.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/

In [32]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_cat.fit(project_data_train['clean_categories'].values) #fitting has to be on Train data

train_categories_one_hot = vectorizer_cat.transform(project_data_train['clean_categories'].values)
cv_categories_one_hot = vectorizer_cat.transform(project_data_cv['clean_categories'].values)
test_categories_one_hot = vectorizer_cat.transform(project_data_test['clean_categories'].values)


print(vectorizer_cat.get_feature_names())
print("Shape of training data matrix after one hot encoding ",train_categories_one_hot.shape)
print("Shape of cross validation data matrix after one hot encoding ",cv_categories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",test_categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'M
usic_Arts', 'AppliedLearning', 'SpecialNeeds',
 'Health_Sports', 'Math_Science', 'Literacy_La
nguage']
Shape of training data matrix after one hot en
coding  (8978, 9)
Shape of cross validation data matrix after on
e hot encoding  (4422, 9)
Shape of test data matrix after one hot encodi
ng  (6600, 9)
```

In [ ]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_d
ict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data_train['clean_subcategories'].valu
es)


train_subcategories_one_hot = vectorizer.transform(project_da
ta_train['clean_subcategories'].values)
cv_subcategories_one_hot = vectorizer.transform(project_data_
cv['clean_subcategories'].values)
test_subcategories_one_hot = vectorizer.transform(project_dat
a_test['clean_subcategories'].values)


print(vectorizer.get_feature_names())


print("Shape of train data matrix after one hot encoding ",tr
ain_subcategories_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_subcategories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_subcategories_one_hot.shape)
```

In [33]:

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


train_subcategories_one_hot = vectorizer_subcat.transform(project_data_train['clean_subcategories'].values)
cv_subcategories_one_hot = vectorizer_subcat.transform(project_data_cv['clean_subcategories'].values)
test_subcategories_one_hot = vectorizer_subcat.transform(project_data_test['clean_subcategories'].values)


print(vectorizer_subcat.get_feature_names())


print("Shape of train data matrix after one hot encoding ",train_subcategories_one_hot.shape)
print("Shape of cross validation data matrix after one hot encoding ",cv_subcategories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",test_subcategories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLi
teracy', 'ParentInvolvement', 'Extracurricular
', 'Civics_Government', 'ForeignLanguages', 'N
utritionEducation', 'Warmth', 'Care_Hunger', '
SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'College_CareerPrep',
 'Other', 'Music', 'History_Geography', 'Healt
h_LifeScience', 'ESL', 'EarlyDevelopment', 'Gy
m_Fitness', 'EnvironmentalScience', 'VisualArt
```

```
s', 'Health_Wellness', 'AppliedSciences', 'Spe
cialNeeds', 'Literature_Writing', 'Mathematics
', 'Literacy']
Shape of train data matrix after one hot encod
ing  (8978, 30)
Shape of cross validation data matrix after on
e hot encoding  (4422, 30)
Shape of test data matrix after one hot encodi
ng  (6600, 30)
```

In [34]:

```
## we use count vectorizer to convert the values into one hot
 encoded features

vectorizer_school_state = CountVectorizer()
vectorizer_school_state.fit(project_data_train['school_state']
.values)



print(vectorizer_school_state.get_feature_names())

train_school_state_category_one_hot = vectorizer_school_state
.transform(project_data_train['school_state'].values)

cv_school_state_category_one_hot = vectorizer_school_state.tr
ansform(project_data_cv['school_state'].values)
test_school_state_category_one_hot = vectorizer_school_state.
transform(project_data_test['school_state'].values)


print("Shape of train data matrix after one hot encoding ",tr
ain_school_state_category_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_school_state_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
```

```
t_school_state_category_one_hot.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc
', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', '
in', 'ks', '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 train data matrix after one hot encod
ing  (8978, 51)
Shape of cross validation data matrix after on
e hot encoding  (4422, 51)
Shape of test data matrix after one hot encodi
ng  (6600, 51)
```

In [35]:

```
#This step is to intialize a vectorizer with vocab from train
 data
my_counter = Counter()
for project_grade in project_data_train['project_grade_catego
ry'].values:
    my_counter.update(project_grade.split())
```

In [36]:

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

In [37]:

```
## we use count vectorizer to convert the values into one hot
 encoded features

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_pro
ject_grade_cat_dict.keys()), lowercase=False, binary=True)
```

```
vectorizer_grade.fit(project_data_train['project_grade_category'].values)



print(vectorizer_grade.get_feature_names())

train_project_grade_category_one_hot = vectorizer_grade.transform(project_data_train['project_grade_category'].values)
cv_project_grade_category_one_hot = vectorizer_grade.transform(project_data_cv['project_grade_category'].values)
test_project_grade_category_one_hot = vectorizer_grade.transform(project_data_test['project_grade_category'].values)


print("Shape of train data matrix after one hot encoding ",train_project_grade_category_one_hot.shape)
print("Shape of cross validation data matrix after one hot encoding ",cv_project_grade_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",test_project_grade_category_one_hot.shape)
```

```
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'Grades_PreK_2']
Shape of train data matrix after one hot encoding  (8978, 4)
Shape of cross validation data matrix after one hot encoding  (4422, 4)
Shape of test data matrix after one hot encoding  (6600, 4)
```

In [38]:

```
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is-an-invalid-document
#ValueError: np.nan is an invalid document, expected byte or unicode string.
vectorizer_prefix = CountVectorizer()



```

```
vectorizer_prefix.fit(project_data_train['teacher_prefix'].values.astype("U"))


print(vectorizer_prefix.get_feature_names())


train_teacher_prefix_categories_one_hot = vectorizer_prefix.transform(project_data_train['teacher_prefix'].values.astype("U"))
cv_teacher_prefix_categories_one_hot = vectorizer_prefix.transform(project_data_cv['teacher_prefix'].values.astype("U"))
test_teacher_prefix_categories_one_hot = vectorizer_prefix.transform(project_data_test['teacher_prefix'].values.astype("U"))

print("Shape of train data matrix after one hot encoding ",train_teacher_prefix_categories_one_hot.shape)
print("Shape of cross validation data matrix after one hot encoding ",cv_teacher_prefix_categories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",test_teacher_prefix_categories_one_hot.shape)
```

```
['mr', 'mrs', 'ms', 'teacher']
Shape of train data matrix after one hot encod
ing  (8978, 4)
Shape of cross validation data matrix after on
e hot encoding  (4422, 4)
Shape of test data matrix after one hot encodi
ng  (6600, 4)
```

2.3 Make Data Model Ready: encoding essay, and project_title

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [39]:

```
# We are considering only the words which appeared in at leas
t 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(project_data_train['preprocessed_ess
ays'].values)  #Fitting has to be on Train data


train_essay_bow = vectorizer_bow_essay.transform(project_data
_train['essay'].values)
cv_essay_bow = vectorizer_bow_essay.transform(project_data_cv
['essay'].values)
test_essay_bow = vectorizer_bow_essay.transform(project_data_
test['essay'].values)


print("Shape of train data matrix after one hot encoding ",tr
ain_essay_bow.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_essay_bow.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_essay_bow.shape)
```

```
Shape of train data matrix after one hot encod
ing  (8978, 5866)
Shape of cross validation data matrix after on
e hot encoding  (4422, 5866)
Shape of test data matrix after one hot encodi
ng  (6600, 5866)
```

In [40]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit_transform(project_data_train['prepro
cessed_titles'].values)    #Fitting has to be on Train data


train_title_bow = vectorizer_bow_title.transform(project_data
_train['preprocessed_titles'].values)
cv_title_bow = vectorizer_bow_title.transform(project_data_cv
['preprocessed_titles'].values)
test_title_bow = vectorizer_bow_title.transform(project_data_
test['preprocessed_titles'].values)



print("Shape of train data matrix after one hot encoding ",tr
ain_title_bow.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_title_bow.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_title_bow.shape)
```

```
Shape of train data matrix after one hot encod
ing  (8978, 623)
Shape of cross validation data matrix after on
e hot encoding  (4422, 623)
Shape of test data matrix after one hot encodi
ng  (6600, 623)
```

1.5.2.2 TFIDF vectorizer

In [41]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(project_data_train['preprocessed_essays'])      #Fitting has to be on Train data

train_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_train['preprocessed_essays'].values)
cv_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_cv['preprocessed_essays'].values)
test_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_test['preprocessed_essays'].values)

print("Shape of train data matrix after one hot encoding ",train_essay_tfidf.shape)
print("Shape of cross validation data matrix after one hot encoding ",cv_essay_tfidf.shape)
print("Shape of test data matrix after one hot encoding ",test_essay_tfidf.shape)
```

```
Shape of train data matrix after one hot encod
ing  (8978, 5866)
Shape of cross validation data matrix after on
e hot encoding  (4422, 5866)
Shape of test data matrix after one hot encodi
ng  (6600, 5866)
```

In [42]:

```
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(project_data_train['preprocessed_titles'])      #Fitting has to be on Train data
```

```
train_title_tfidf = vectorizer_tfidf_title.transform(project_
data_train['preprocessed_titles'].values)
cv_title_tfidf = vectorizer_tfidf_title.transform(project_dat
a_cv['preprocessed_titles'].values)
test_title_tfidf = vectorizer_tfidf_title.transform(project_d
ata_test['preprocessed_titles'].values)

print("Shape of train data matrix after one hot encoding ",tr
ain_title_tfidf.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_title_tfidf.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_title_tfidf.shape)
```

```
Shape of train data matrix after one hot encod
ing  (8978, 623)
Shape of cross validation data matrix after on
e hot encoding  (4422, 623)
Shape of test data matrix after one hot encodi
ng  (6600, 623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [43]:

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

In [44]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_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
    train_avg_w2v_essays.append(vector)

print(len(train_avg_w2v_essays))
print(len(train_avg_w2v_essays[0]))
```

```
100%|██████████| 8978/8978 [00:01<00:00, 5808.95it/s]
```

```
8978
300
```

In [45]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essays']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
```

```
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    cv_avg_w2v_essays.append(vector)

print(len(cv_avg_w2v_essays))
print(len(cv_avg_w2v_essays[0]))
```

```
100%|██████████| 4422/4422 [00:00<00:00, 5630.
84it/s]
```

```
4422
300
```

In [46]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_essays = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essays']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
```

```
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_avg_w2v_essays.append(vector)

print(len(test_avg_w2v_essays))
print(len(test_avg_w2v_essays[0]))
```

```
100%|██████████| 6600/6600 [00:01<00:00, 5680.
87it/s]
```

```
6600
300
```

In [47]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_titles = []; # the avg-w2v for each sentence/re
view is stored in this list
for sentence in tqdm(project_data_train['preprocessed_titles'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_avg_w2v_titles.append(vector)

print(len(train_avg_w2v_titles))
print(len(train_avg_w2v_titles[0]))
```

```
100%|██████████| 8978/8978 [00:00<00:00, 93621
.09it/s]
```

```
8978
300
```

In [48]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_avg_w2v_titles = []; # the avg-w2v for each sentence/revie
w is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_titles']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/review
    for word in sentence.split(): # for each word in a review
/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    cv_avg_w2v_titles.append(vector)

print(len(cv_avg_w2v_titles))
print(len(cv_avg_w2v_titles[0]))
```

```
100%|██████████| 4422/4422 [00:00<00:00, 89125
.58it/s]
```

```
4422
300
```

In [49]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_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
    test_avg_w2v_titles.append(vector)

print(len(test_avg_w2v_titles))
print(len(test_avg_w2v_titles[0]))
```

```
100%|██████████| 6600/6600 [00:00<00:00, 90474.84it/s]
```

```
6600
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [50]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_essays'].val
```

```
ues)
# 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(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [51]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_essays = []; # the avg-w2v for each sentence/
review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_essays'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_essays.append(vector)

print(len(train_tfidf_w2v_essays))
```

```
print(len(train_tfidf_w2v_essays[0]))
```

```
100%|██████████| 8978/8978 [00:12<00:00, 735.3
8it/s]
```

```
8978
300
```

In [52]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_tfidf_w2v_essays = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essays']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_tfidf_w2v_essays.append(vector)
```

```
print(len(cv_tfidf_w2v_essays))
print(len(cv_tfidf_w2v_essays[0]))
```

```
100%|██████████| 4422/4422 [00:05<00:00, 762.7
1it/s]
```

```
4422
300
```

In [53]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_essays = []; # the avg-w2v for each sentence/r
eview is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essays']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_essays.append(vector)
```

```
print(len(test_tfidf_w2v_essays))
print(len(test_tfidf_w2v_essays[0]))
```

```
100%|██████████| 6600/6600 [00:08<00:00, 760.5
1it/s]
```

```
6600
300
```

In [54]:

```
# Similarly you can vectorize for title also
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_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(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [55]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_titles = []; # the avg-w2v for each sentence/
review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_titles'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
```

```
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_titles.append(vector)

print(len(train_tfidf_w2v_titles))
print(len(train_tfidf_w2v_titles[0]))
```

```
100%|██████████| 8978/8978 [00:00<00:00, 42066.55it/s]
```

```
8978
300
```

In [56]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_tfidf_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_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 w
```

```
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_tfidf_w2v_titles.append(vector)

print(len(cv_tfidf_w2v_titles))
print(len(cv_tfidf_w2v_titles[0]))
```

```
100%|██████████| 4422/4422 [00:00<00:00, 43945
.12it/s]
```

```
4422
300
```

In [57]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_titles = []; # the avg-w2v for each sentence/r
eview is stored in this list
for sentence in tqdm(project_data_test['preprocessed_titles']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_titles.append(vector)

print(len(test_tfidf_w2v_titles))
print(len(test_tfidf_w2v_titles[0]))
```

```
100%|██████████| 6600/6600 [00:00<00:00, 44259
.70it/s]
```

```
6600
300
```

1.5.3 Vectorizing Numerical features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [58]:

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

In [59]:

```
project_data_train = pd.merge(project_data_train, price_data,
 on='id', how='left')
project_data_cv = pd.merge(project_data_cv, price_data, on='i
d', how='left')
project_data_test = pd.merge(project_data_test, price_data, o
n='id', how='left')
```

In [60]:

```
from sklearn.preprocessing import Normalizer
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer = Normalizer()
normalizer.fit(project_data_train['price'].values.reshape(-1,1
))


price_normalized_train = normalizer.transform(project_data_tr
ain['price'].values.reshape(-1, 1))
price_normalized_cv = normalizer.transform(project_data_cv['p
rice'].values.reshape(-1, 1))
price_normalized_test = normalizer.transform(project_data_tes
t['price'].values.reshape(-1, 1))


print('After normalization')
print(price_normalized_train.shape)
print(price_normalized_cv.shape)
print(price_normalized_test.shape)
```

```
After normalization
(8978, 1)
```

```
(4422, 1)
(6600, 1)
```

In [61]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))


# Now standardize the data with above maen and variance.
previously_posted_projects_normalized_train = normalizer.transform(project_data_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
previously_posted_projects_normalized_cv = normalizer.transform(project_data_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
previously_posted_projects_normalized_test = normalizer.transform(project_data_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))


print('After normalization')
print(previously_posted_projects_normalized_train.shape)
print(previously_posted_projects_normalized_cv.shape)
print(previously_posted_projects_normalized_test.shape)
```

```
After normalization
(8978, 1)
(4422, 1)
(6600, 1)
```

Assignment 3: Apply KNN

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

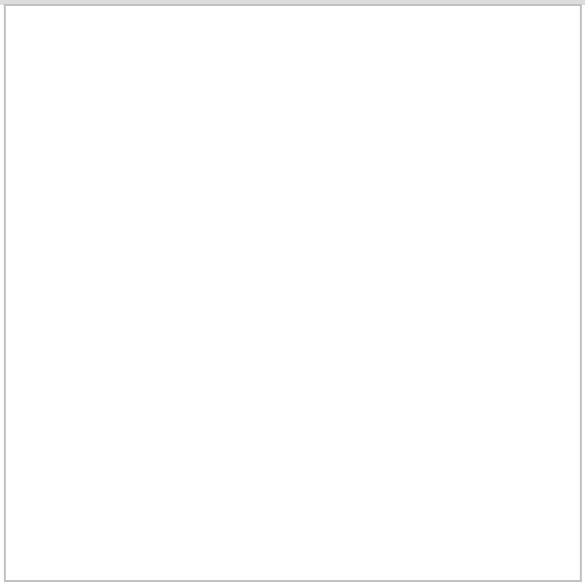
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)
    - Set 4: categorical,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

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

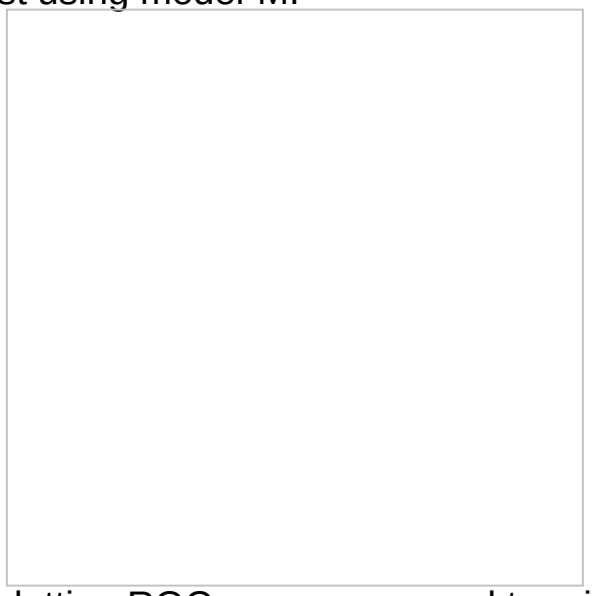
    - Find the best hyper parameter which results in the maximum AUC value
    - Find the best hyper paramter using k-fold cross validation (or) simple cross validation data
    - Use gridsearch-cv or randomsearch-cv or write your own for loops to do this task

3. **Representation of results**

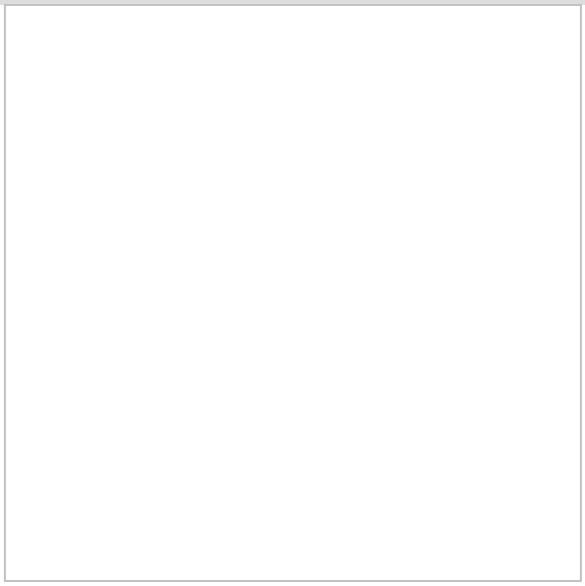
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure



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



- Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points



4. **[Task-2]**

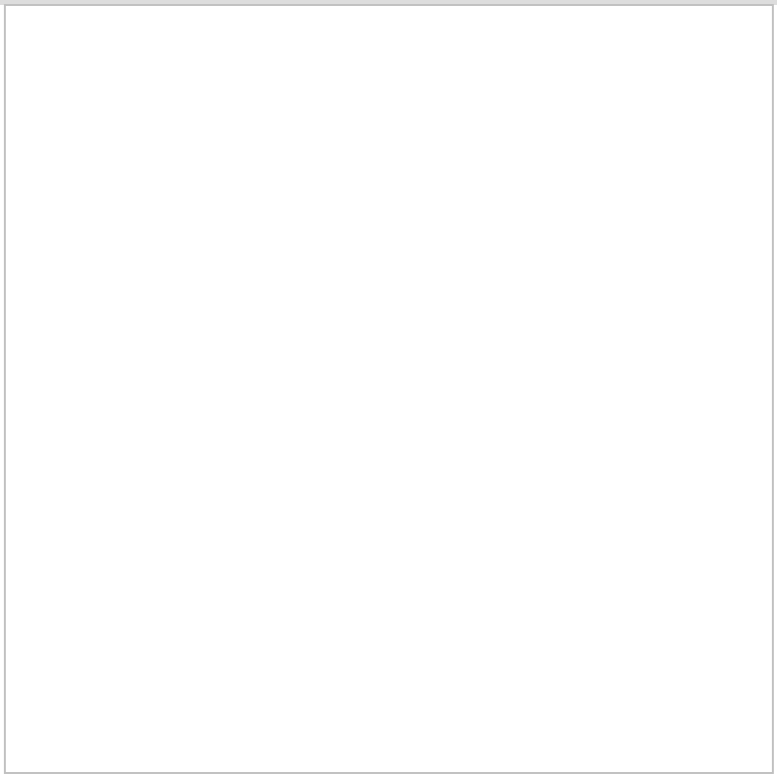
   - Select top 2000 features from feature Set 2 using `SelectKBest` and then apply KNN on top of these features

   - ```
     from sklearn.datasets import load_digits
     from sklearn.feature_selection import SelectKBest, chi2
     X, y = load_digits(return_X_y=True)
     X.shape
     X_new = SelectKBest(chi2, k=20).fit_transform(X, y)
     X_new.shape
     ========
     output:
     (1797, 64)
     (1797, 20)
     ```

   - Repeat the steps 2 and 3 on the data matrix after feature selection

5. **Conclusion**

   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link
     ```



Note: Data Leakage

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

2. K Nearest Neighbor

2.4 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

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

In [62]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_bow, train_title_bow, train_school_state_category_one_hot,train_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_train, train_project_grade_category_one_hot, price_normalized_train)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_essay_bow, cv_title_bow, cv_school_state_category_one_hot, cv_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_cv, cv_project_grade_category_one_hot, price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_bow, test_title_bow, test_school_state_ca
```

```
tegory_one_hot, test_teacher_prefix_categories_one_hot, previ
ously_posted_projects_normalized_test, test_project_grade_cat
egory_one_hot, price_normalized_test)).tocsr()
```

In [63]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(8978, 6589)
(4422, 6589)
(6600, 6589)
```

In [64]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop wil
l be 49041 - 49041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 mul
tiplier
    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 [65]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier()

parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51, 65, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv= 5, scoring='roc_auc')

clf.fit(X_train, y_train)

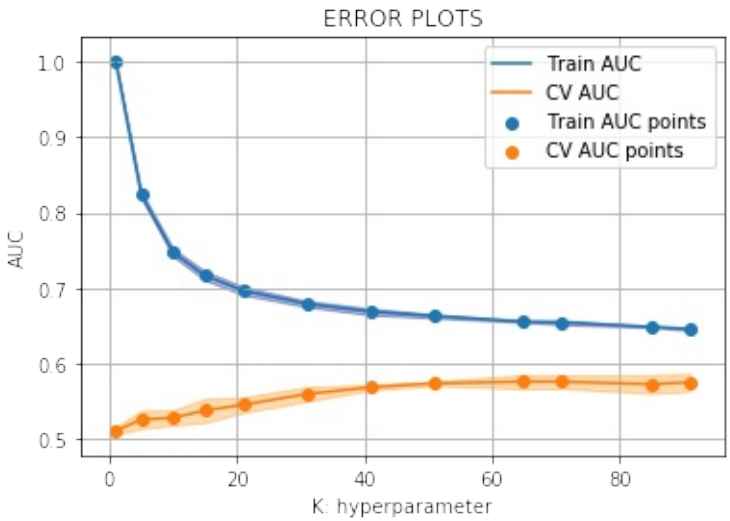
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['n_neighbors'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.3,color='darkblue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorange')

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

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



In [66]:

```
#https://datascience.stackexchange.com/questions/21877/how-to-use-the-output-of-gridsearch
#choosing the best hyperparameter
clf.best_params_
```

Out[66]:

```
{'n_neighbors': 65}
```

Train the model using the best hyperparameter that will maximise AUC score using GridSearchCV

In [67]:

```
best_k1 = clf.best_params_['n_neighbors']
```

In [68]:

```
best_k1
```

Out[68]:

65

In [69]:

```
# 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

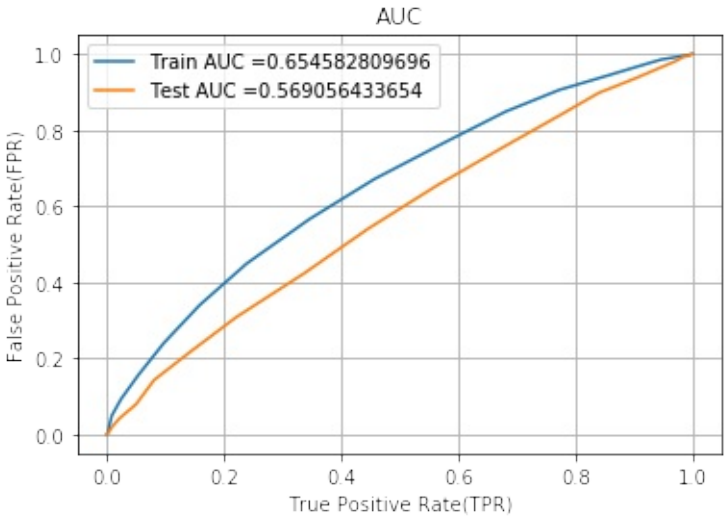
neigh = KNeighborsClassifier(n_neighbors=best_k1)
neigh.fit(X_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_train)
y_test_pred = batch_predict(neigh, X_test)

train_fpr, train_tpr, train_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, test_thresholds = roc_curve(y_test, y_tes
```

```
t_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [70]:

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

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

In [71]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(train_thresholds, train_fpr, tra
in_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.37031326844
```

```
6 for threshold 0.831
Train confusion matrix
[[ 895  474]
 [3299 4310]]
```

In [72]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

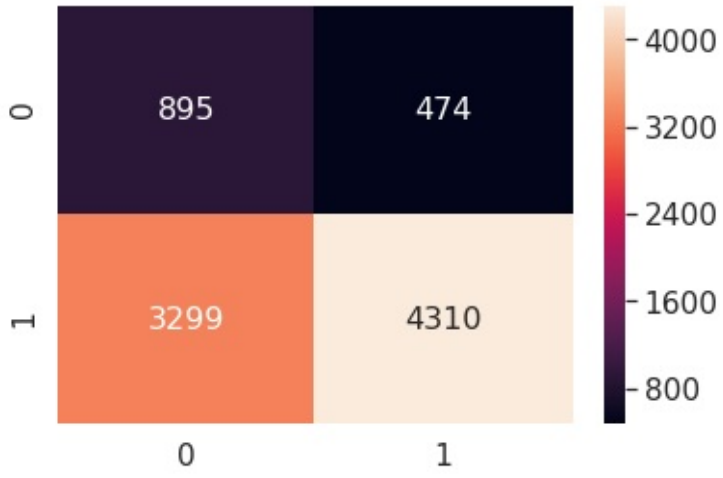
print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[72]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbef70c9f98>
```



In [73]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 558  448]
 [2573 3021]]
```

In [74]:

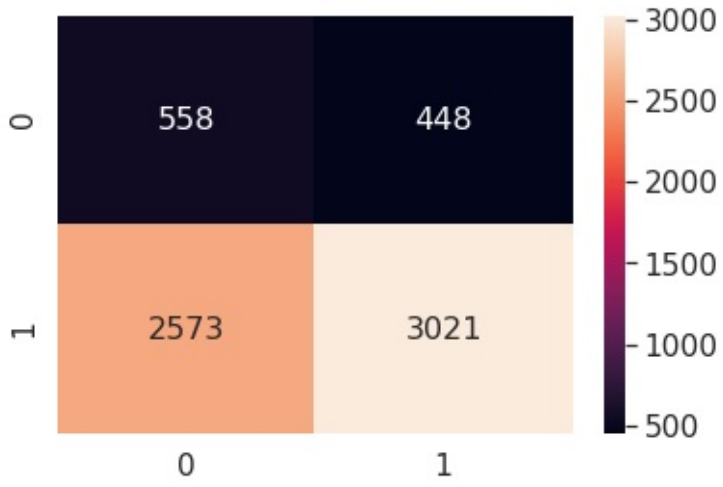
```
print("Test data confusion matrix")

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

```
Test data confusion matrix
```

Out[74]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbef6fa7828>
```



Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_essay (TFIDF)

In [75]:

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

X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_tfidf, train_title_tfidf, train_school_state_category_one_hot,train_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_train, train_project_grade_category_one_hot, price_normalized_train)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_essay_tfidf, cv_title_tfidf, cv_school_state_category_one_hot, cv_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_cv, cv_project_grade_category_one_hot,
 price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_tfidf, test_title_tfidf, test_school_state_category_one_hot, test_teacher_prefix_categories_one_hot, previously_posted_projects_normalized_test, test_project_grade_category_one_hot, price_normalized_test)).tocsr()
```

In [76]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(8978, 6589)
(4422, 6589)
(6600, 6589)
```

In [77]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
```

```
odel_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier()

parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51, 65
, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv= 5, scoring='roc_auc
')

clf.fit(X_train, y_train)

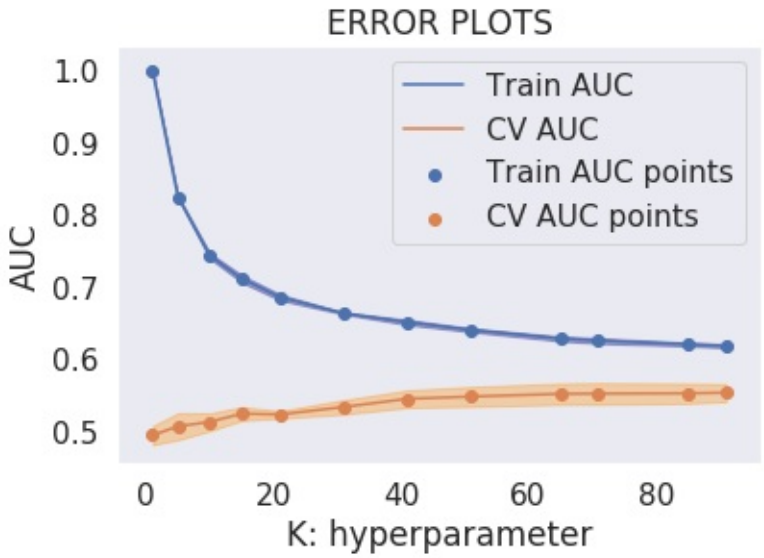
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['n_neighbors'], train_auc, label='Train A
UC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc -
train_auc_std,train_auc + train_auc_std,alpha=0.3,color='dark
blue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_
auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorange')

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

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



In [78]:

```
#https://datascience.stackexchange.com/questions/21877/how-to
-use-the-output-of-gridsearch
#choosing the best hyperparameter
clf.best_params_
```

Out[78]:

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

Train the model using the best hyperparameter that will maximise AUC score using GridSearchCV

In [79]:

```
best_k2 = clf.best_params_['n_neighbors']
```

In [80]:

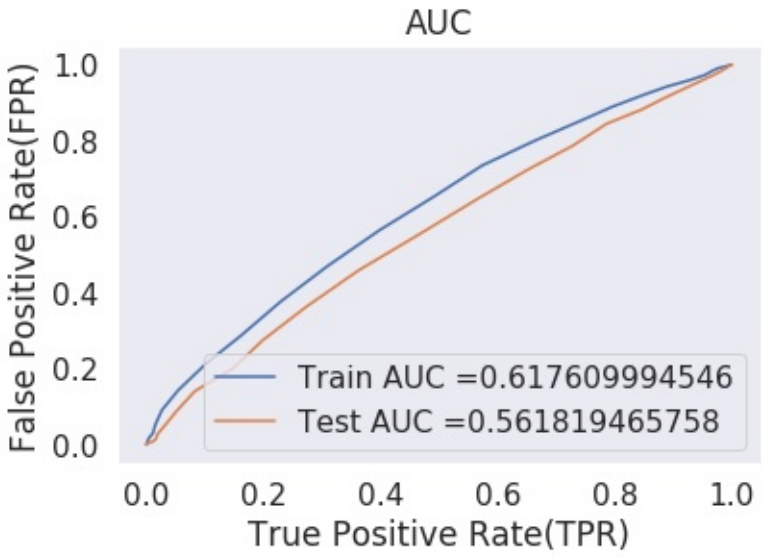
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k2)
neigh.fit(X_train, 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_train)
y_test_pred = batch_predict(neigh, X_test)

train_fpr, train_tpr, train_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, test_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("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [81]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(train_thresholds, train_fpr, tra
in_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.33959170069
 for threshold 0.857
Train confusion matrix
[[ 824  545]
 [3316 4293]]
```

In [82]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
```

```
-a-confusion-matrix

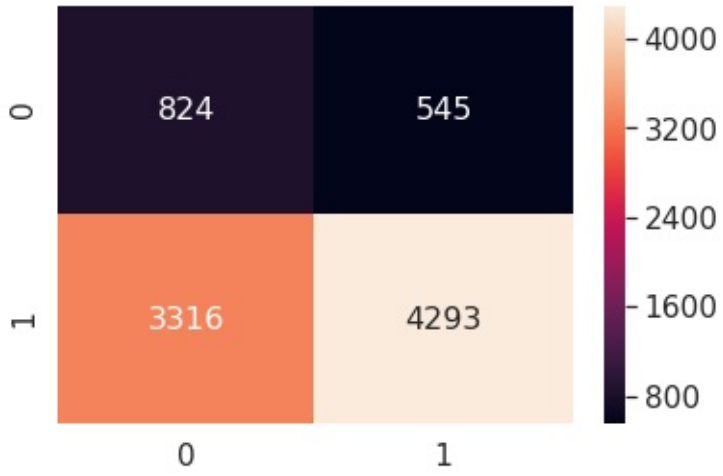
print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[82]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
bef6fb5438>
```



In [83]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 537  469]
 [2497 3097]]
```

In [84]:

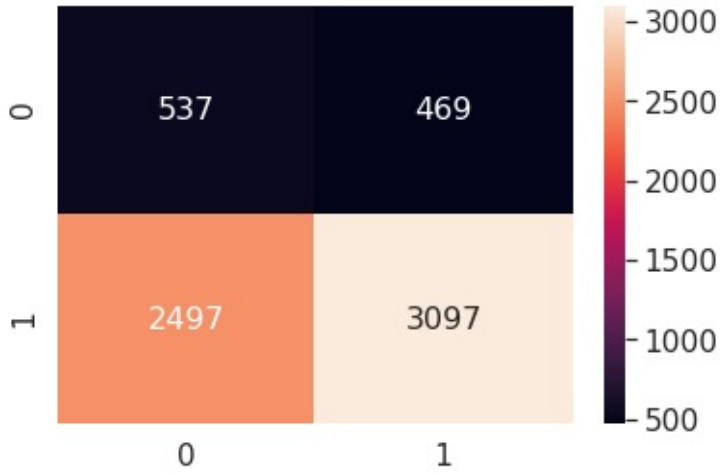
```
print("Test data confusion matrix")

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

```
Test data confusion matrix
```

Out[84]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
beff829668>
```



Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_essay (AVG W2V)

In [85]:

```
# merge two sparse matrices: https://stackoverflow.com/a/1971
```

```
0648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse
 matrix and a dense matirx :)
X_train = hstack((train_categories_one_hot, train_subcategori
es_one_hot, train_avg_w2v_essays, train_avg_w2v_titles, train
_school_state_category_one_hot,train_teacher_prefix_categorie
s_one_hot, previously_posted_projects_normalized_train, train
_project_grade_category_one_hot, price_normalized_train)).toc
sr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_ho
t, cv_avg_w2v_essays, cv_avg_w2v_titles, cv_school_state_cate
gory_one_hot, cv_teacher_prefix_categories_one_hot, previousl
y_posted_projects_normalized_cv, cv_project_grade_category_on
e_hot, price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_
one_hot, test_avg_w2v_essays, test_avg_w2v_titles, test_schoo
l_state_category_one_hot, test_teacher_prefix_categories_one_
hot, previously_posted_projects_normalized_test, test_project
_grade_category_one_hot, price_normalized_test)).tocsr()
```

In [86]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(8978, 700)
(4422, 700)
(6600, 700)
```

In [87]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
odel_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier()
```

```
parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51, 65
, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv= 5, scoring='roc_auc
')

clf.fit(X_train, y_train)

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

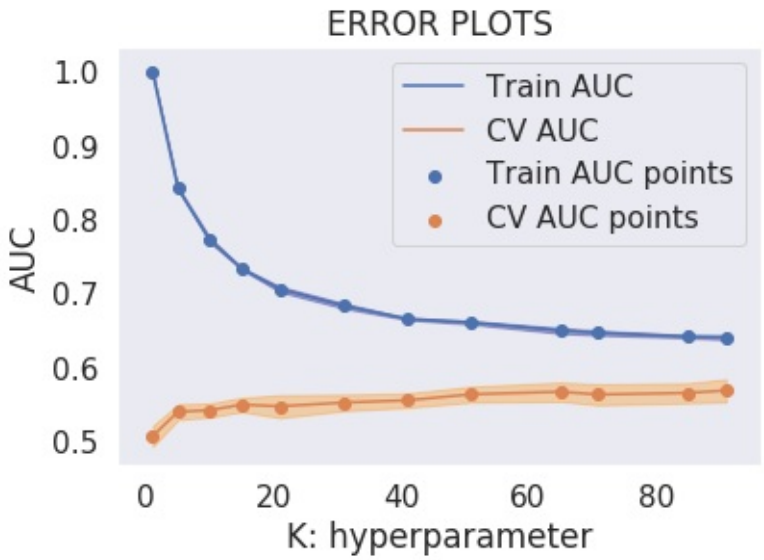
plt.plot(parameters['n_neighbors'], train_auc, label='Train A
UC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc -
train_auc_std,train_auc + train_auc_std,alpha=0.3,color='dark
blue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_
auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorange')

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


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
```

```
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [88]:

```
#https://datascience.stackexchange.com/questions/21877/how-to
-use-the-output-of-gridsearch
#choosing the best hyperparameter
clf.best_params_
```

Out[88]:

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

Train the model using the best hyperparameter that will maximise AUC score using GridSearchCV

In [89]:

```
best_k3 = clf.best_params_['n_neighbors']
```

In [90]:

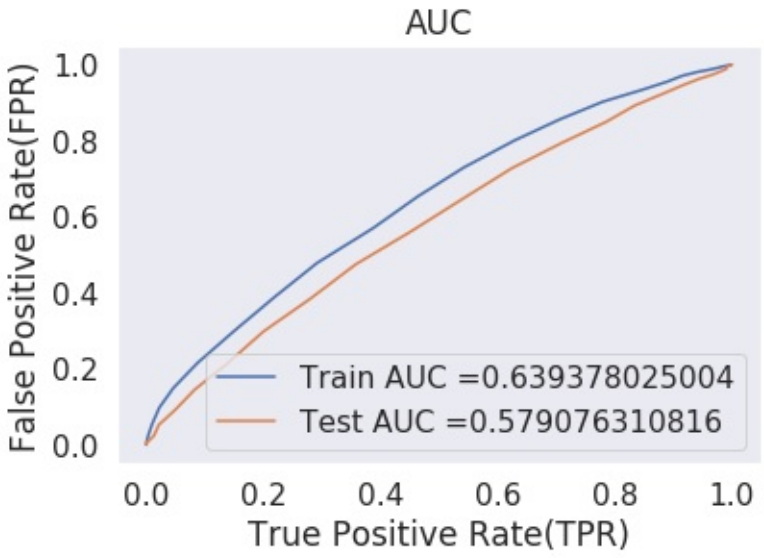
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve


neigh = KNeighborsClassifier(n_neighbors=best_k3)
neigh.fit(X_train, 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_train)
y_test_pred = batch_predict(neigh, X_test)

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

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



Confusion Matrix of Train and Test Data

In [91]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.35027471696
7 for threshold 0.846
Train confusion matrix
[[ 734  635]
 [2638 4971]]
```

In [92]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
```

```
-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[92]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
bef45c3a90>
```

In [93]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 461  545]
 [1968 3626]]
```

In [94]:

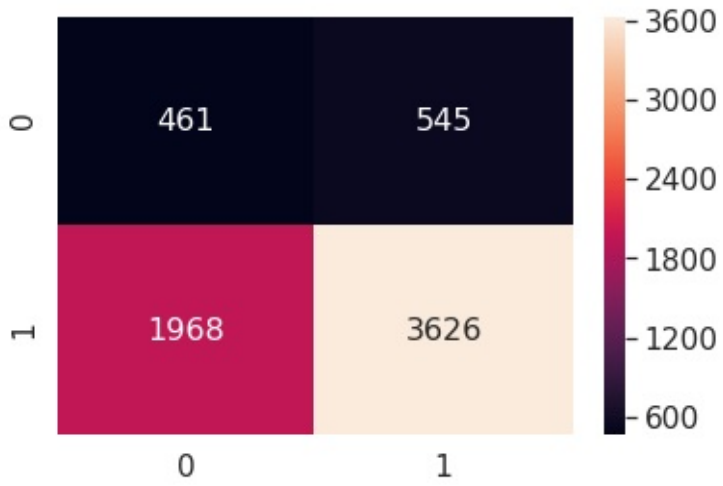
```
print("Test data confusion matrix")

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

```
Test data confusion matrix
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
bef6f352b0>
```



Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

In [95]:

```
# merge two sparse matrices: https://stackoverflow.com/a/1971
```

```
0648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse
 matrix and a dense matirx :)
X_train = hstack((train_categories_one_hot, train_subcategori
es_one_hot, train_tfidf_w2v_essays, train_tfidf_w2v_titles, t
rain_school_state_category_one_hot,train_teacher_prefix_categ
ories_one_hot, previously_posted_projects_normalized_train, t
rain_project_grade_category_one_hot, price_normalized_train))
.tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_ho
t, cv_tfidf_w2v_essays, cv_tfidf_w2v_titles, cv_school_state_
category_one_hot, cv_teacher_prefix_categories_one_hot, previ
ously_posted_projects_normalized_cv, cv_project_grade_categor
y_one_hot, price_normalized_cv)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_
one_hot, test_tfidf_w2v_essays, test_tfidf_w2v_titles, test_s
chool_state_category_one_hot, test_teacher_prefix_categories_
one_hot, previously_posted_projects_normalized_test, test_pro
ject_grade_category_one_hot, price_normalized_test)).tocsr()
```

In [96]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(8978, 700)
(4422, 700)
(6600, 700)
```

In [97]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
odel_selection.GridSearchCV.html

neigh = KNeighborsClassifier()
```

```
parameters = {'n_neighbors':[1, 5, 10, 15, 21, 31, 41, 51, 65
, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv= 5, scoring='roc_auc
')

clf.fit(X_train, y_train)

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

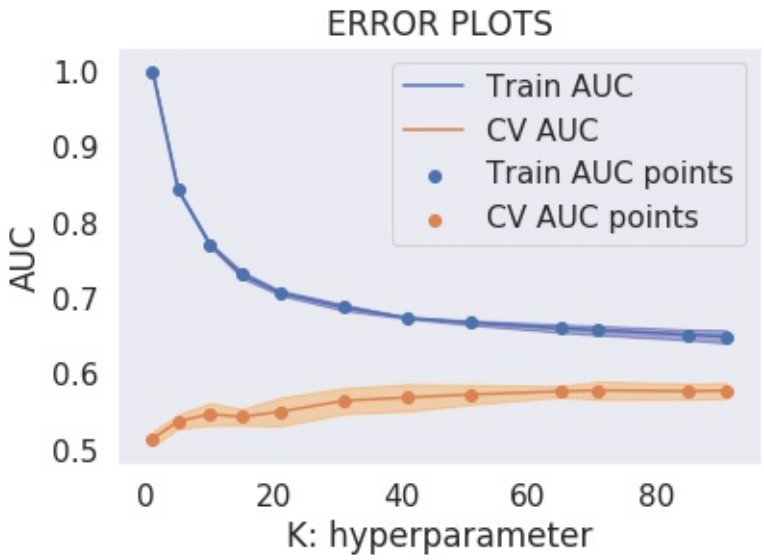
plt.plot(parameters['n_neighbors'], train_auc, label='Train A
UC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],train_auc -
train_auc_std,train_auc + train_auc_std,alpha=0.3,color='dark
blue')

plt.plot(parameters['n_neighbors'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/
48803361/4084039
plt.gca().fill_between(parameters['n_neighbors'],cv_auc - cv_
auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorange')

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


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

```
plt.grid()
plt.show()
```



In [98]:

```
#https://datascience.stackexchange.com/questions/21877/how-to
-use-the-output-of-gridsearch
#choosing the best hyperparameter
clf.best_params_
```

Out[98]:

```
{'n_neighbors': 71}
```

Train the model using the best hyperparameter that will maximise AUC score using GridSearchCV

In [99]:

```
best_k4 = clf.best_params_['n_neighbors']
```

In [101]:

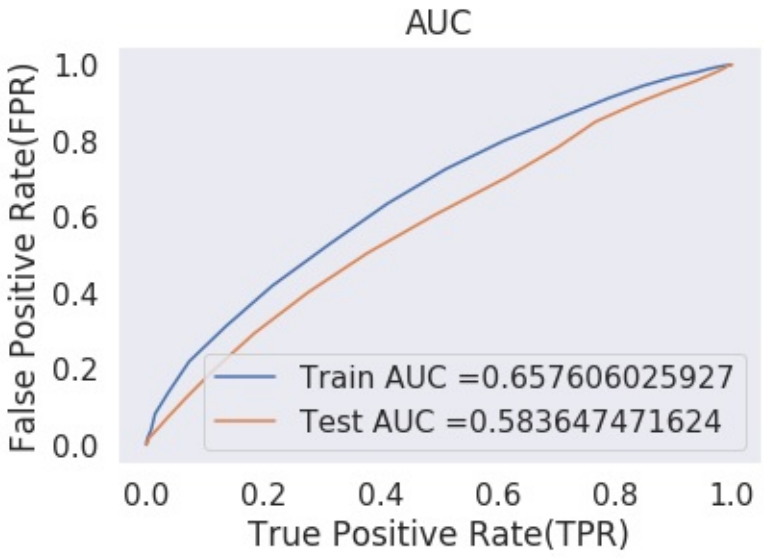
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve


neigh = KNeighborsClassifier(n_neighbors=best_k4)
neigh.fit(X_train, 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_train)
y_test_pred = batch_predict(neigh, X_test)

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

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



Confusion Matrix of Train and Test Data

In [102]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.37318250148
 for threshold 0.845
Train confusion matrix
[[ 806  563]
 [2786 4823]]
```

In [103]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
```

```
-a-confusion-matrix

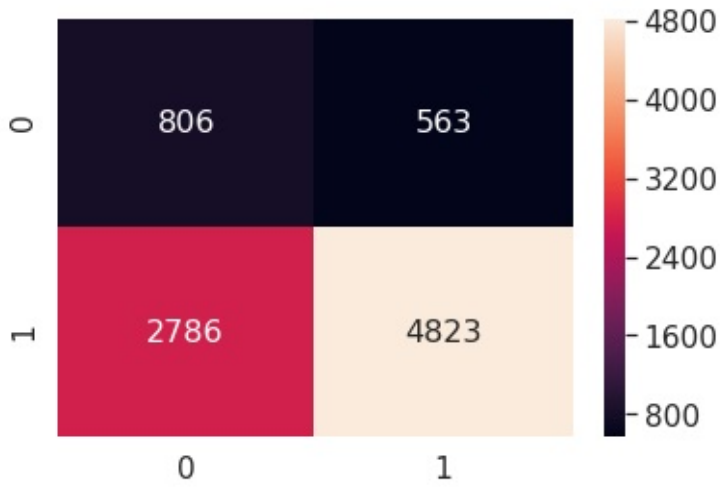
print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[103]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
bef462d710>
```



In [104]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 512  494]
 [2220 3374]]
```

In [105]:

```
print("Test data confusion matrix")

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

```
Test data confusion matrix
```

Out[105]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
bef448d550>
```

Set 2: Feature selection with `SelectKBest`

In [106]:

```
X_train = hstack((train_categories_one_hot, train_subcategori
es_one_hot, train_school_state_category_one_hot, train_projec
t_grade_category_one_hot, train_teacher_prefix_categories_one
_hot, price_normalized_train ,  previously_posted_projects_no
rmalized_train,  train_essay_tfidf, train_title_tfidf)).tocsr
()
X_test = hstack((test_categories_one_hot, test_subcategories_
one_hot, test_school_state_category_one_hot, test_project_gra
de_category_one_hot, test_teacher_prefix_categories_one_hot,
price_normalized_test,  previously_posted_projects_normalized
_test,  test_essay_tfidf, test_title_tfidf)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_ho
t, cv_school_state_category_one_hot, cv_project_grade_categor
y_one_hot, cv_teacher_prefix_categories_one_hot, price_normal
ized_cv,previously_posted_projects_normalized_cv,  cv_essay_t
fidf, cv_title_tfidf)).tocsr()
```

In [107]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(8978, 6589)
(4422, 6589)
(6600, 6589)
```

In [109]:

```
from sklearn.feature_selection import SelectKBest, chi2
#https://scikit-learn.org/stable/modules/generated/sklearn.fe
```

```
ature_selection.SelectKBest.html
selector = SelectKBest(chi2, k = 2000 )
selector.fit(X_train, y_train)
X_train_new = SelectKBest(chi2, k=2000).fit_transform(X_train
, y_train)
X_test_new = SelectKBest(chi2, k=2000).fit_transform(X_test,
y_test)
X_cv_new = SelectKBest(chi2, k=2000).fit_transform(X_cv, y_cv
)
```

In [110]:

```
print(X_train_new.shape, y_train.shape)
print(X_cv_new.shape, y_cv.shape)
print(X_test_new.shape, y_test.shape)
```

```
(8978, 2000) (8978,)
(4422, 2000) (4422,)
(6600, 2000) (6600,)
```

In [111]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
odel_selection.GridSearchCV.html

neigh = KNeighborsClassifier()

parameters = {'n_neighbors':[1, 5, 10, 15, 21, 33, 41, 51, 65
, 71, 85, 91]}

clf = GridSearchCV(neigh, parameters, cv=5, scoring='roc_auc'
)
clf.fit(X_train_new, y_train)

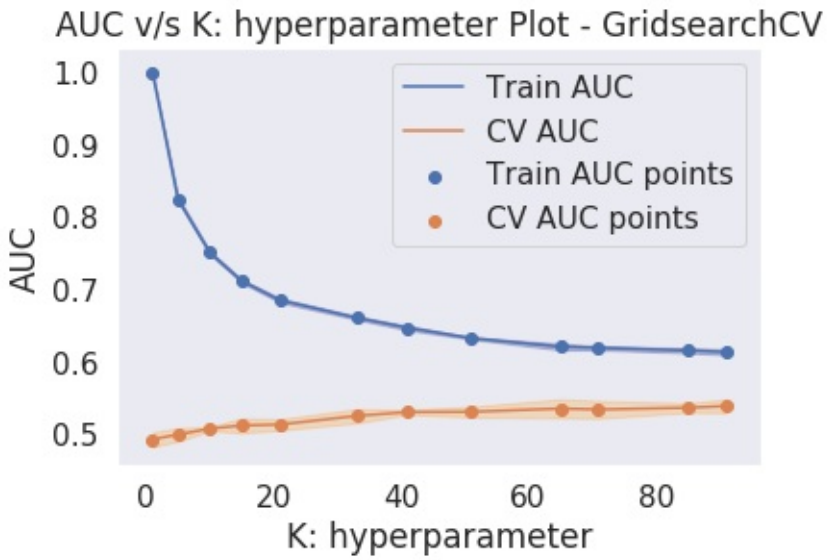
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

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

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

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


plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("AUC v/s K: hyperparameter Plot - GridsearchCV")
plt.grid()
plt.show()
```



In [112]:

```
#https://datascience.stackexchange.com/questions/21877/how-to
-use-the-output-of-gridsearch
#choosing the best hyperparameter
clf.best_params_
```

Out[112]:

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

Train the model using the best hyperparameter that will maximise AUC score using GridSearchCV

In [113]:

```
best_k5 = clf.best_params_['n_neighbors']
```

In [115]:

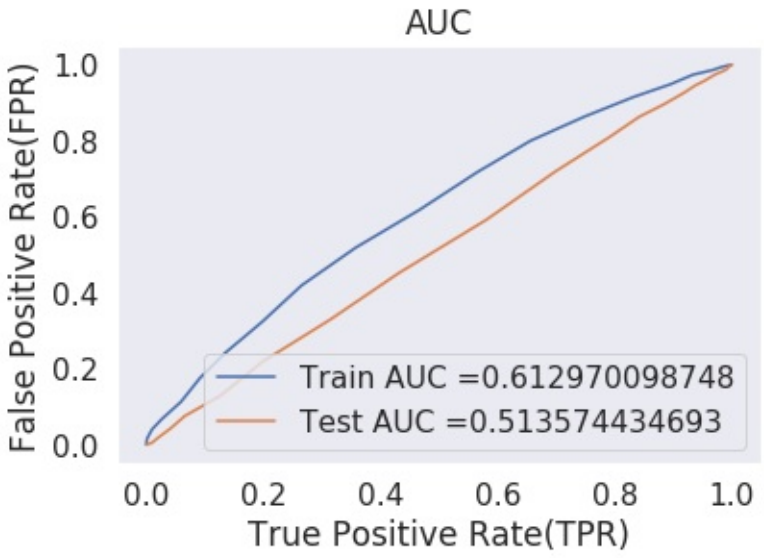
```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
etrics.roc_curve.html#sklearn.metrics.roc_curve
```

```
neigh = KNeighborsClassifier(n_neighbors=best_k5)
neigh.fit(X_train_new, 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_train_new)
y_test_pred = batch_predict(neigh, X_test_new)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_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(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [116]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.33314629430
9 for threshold 0.857
Train confusion matrix
[[ 879  490]
 [3661 3948]]
```

In [117]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
```

```
-a-confusion-matrix

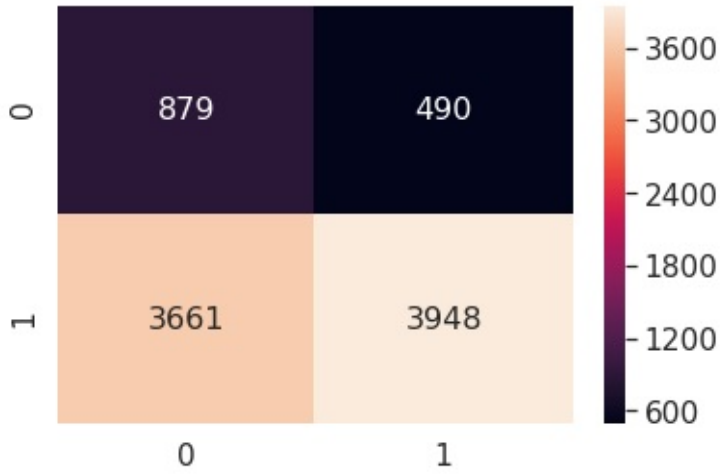
print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[117]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
bedd2b2748>
```



In [118]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 690  316]
 [3752 1842]]
```

In [119]:

```
print("Test data confusion matrix")

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

```
Test data confusion matrix
```

Out[119]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbef4457828>
```

Conclusions

In [120]:

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

# Compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "Test-AUC"]


x.add_row(["BOW", "Brute Force KNN", 65, 0.56])
x.add_row(["TFIDF", "Brute Force KNN", 91, 0.56])
x.add_row(["AVG W2V", "Brute Force KNN", 91, 0.57])
x.add_row(["TFIDF W2V", "Brute Force KNN", 71, 0.58])
x.add_row(["TFIDF", "Feature selection with SelectKBest(Top2000)", 91, 0.51])

print(x)
```

```
+------------+---------------------------------
-------------+-----------------+----------+
| Vectorizer |                      Model
             | Hyper Parameter | Test-AUC |
+------------+---------------------------------
-------------+-----------------+----------+
|    BOW     |                Brute Force KNN
             |        65       |   0.56   |
|   TFIDF    |                Brute Force KNN
```

```
              |        91       |   0.56   |
|  AVG W2V    |                Brute Force KNN
              |        91       |   0.57   |
| TFIDF W2V   |                Brute Force KNN
              |        71       |   0.58   |
|   TFIDF     | Feature selection with SelectKB
est(Top2000) |        91       |   0.51   |
+-------------+---------------------------------
-------------+-----------------+----------+
```