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.

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.

Applied ai course and stackoverflow helped me to do this project

Here I am working on 50000 dataset.

In [2]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

In [3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client
_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&
redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2F
www.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fau
th%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%
20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=
code

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

https://stackoverflow.com/questions/48376580/google-colab-how-to-read-data-from-my-google-drive/52300696 (https://stackoverflow.com/questions/48376580/google-colab-how-to-read-data-from-my-google-drive/52300696)

In [4]:
```
!ls "/content/drive/My Drive"
```

```
'06 Implement SGD.ipynb'              data
 10_DonorsChoose_Clustering.ipynb     glove_vectors
 11_DonorsChoose_TruncatedSVD.ipynb   haberman.csv
 2_DonorsChoose_EDA_TSNE.html         haberman.xlsx
 2_DonorsChoose_EDA_TSNE.ipynb        heat_map.JPG
 2letterstabbrev.pdf                  imdb.txt
 3d_plot.JPG                          navneetkumar384@gmail.com_knn.ipynb
 3d_scatter_plot.ipynb                NavneetMLFormat2.pdf
 4_DonorsChoose_NB.ipynb              navneet_Tsne_assign2.ipynb
 5_DonorsChoose_LR_processed.ipynb    resources.csv
 7_DonorsChoose_SVM.ipynb             response.JPG
 8_DonorsChoose_DT.ipynb              summary.JPG
 9_DonorsChoose_RF_GBDT.ipynb         test_data.csv
 Assignment_SAMPLE_SOLUTION.ipynb     train_cv_auc.JPG
'Assignment_tips(1).docx'             train_data.csv
 Assignment_tips.docx                 train_test_auc.JPG
'Colab Notebooks'                     Untitled0.ipynb
 confusion_matrix.png                 Untitled1.ipynb
 cooc.JPG
```

1.1 Reading Data

In [0]:
```
project_data = pd.read_csv('/content/drive/My Drive/train_data.csv')
resource_data = pd.read_csv('/content/drive/My Drive/resources.csv')
```

In [6]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

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

In [7]:
```
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[7]:

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

In [8]:
```
project_data=project_data.head(50000)
project_data.shape
```

Out[8]: (50000, 17)

In [9]:
```
resource_data=resource_data.head(50000)
resource_data.shape
```

Out[9]: (50000, 4)

In [10]:
```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ["Date" if x=="project_submitted_datetime" else x for x in list(project_data.columns)]

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

# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data=project_data[cols]
project_data.head(2)
```

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	D
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	20 04 00:53
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	20 04 01:05

In [11]:
```
print("Number of datapoints in train value", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of datapoints in train value (50000, 4)
['id' 'description' 'quantity' 'price']
```

Out[11]:

	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

Merging Two dataframes

In [0]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [0]:
```
#https://stackoverflow.com/questions/18689823/pandas-dataframe-replace-nan-values-with-average-of-columns/18691949
#Filling the nan value of price and quantity with mean if any
project_data['price'] = project_data['price'].fillna((project_data['price'].mean()))
project_data['quantity'] = project_data['quantity'].fillna((project_data['quantity'].mean()))
```

1.2 preprocessing of project_subject_categories

In [0]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & 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('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of project_subject_subcategories

In [0]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "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 ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.4 preprocessing of `teacher prefix

In [0]:
```
#project_data.teacher_prefix.replace(-1, np.nan)  #https://stackoverflow.com/questions/41882011/pandas-handling-nans-in-categorical-data
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-attribute-split
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna("") #fill all NaN value with ""
prefix_teacher = list(project_data['teacher_prefix'].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

teacher_prefix_list = []
for i in prefix_teacher:
    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('&','_')
    teacher_prefix_list.append(temp.strip())

project_data['teach_pref'] = teacher_prefix_list #create new column having name teach_pref with preprocessed data
project_data.drop(['teacher_prefix'], axis=1, inplace=True) #delete the teacher_prefix column

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

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

1.5 preprocessing of `project_grade_category

In [0]:
```
#project_data.project_grade_category.replace(-1, np.nan)  #https://stackoverflow.com/questions/41882011/pandas-handling-nans-in-categorical-data
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-attribute-split
project_data['project_grade_category']=project_data['project_grade_category'].fillna("") #fill all NaN value with ""
project_grad_cat = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

project_grad_list = []
for i in project_grad_cat:
    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('&','_')
    project_grad_list.append(temp.strip())

project_data['project_grad_cat'] = project_grad_list    #create new column having name project_grad_cat with preprocessed data
project_data.drop(['project_grade_category'], axis=1, inplace=True) #delete the project_grade_category column

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grad_dict = dict(my_counter)  #this will make a dictionary with keys and values of words and its counts
sorted_project_grad_dict = dict(sorted(project_grad_dict.items(), key=lambda kv: kv[1])) #result a sorted dictionary by number of counts
```

1.6 preprocessing of project_subject_subcategories

In [0]:
```
#project_data.project_grade_category.replace(-1, np.nan)  #https://stackoverflow.com/questions/41882011/pandas-handling-nans-in-categorical-data
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-attribute-split
project_data['school_state']=project_data['school_state'].fillna("") #fill all NaN value with ""
project_school_state = list(project_data['school_state'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

project_school_state_list = []
for i in project_school_state:
    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 traihowever spaces
        temp = temp.replace('&','_')
    project_school_state_list.append(temp.strip())

project_data['project_school_state'] = project_school_state_list    #create new column having name project_grad_cat with preprocessed data
project_data.drop(['school_state'], axis=1, inplace=True) #delete the project_grade_category column

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_school_state_dict = dict(my_counter)  #this will make a dictionary with keys and values of words and its counts
sorted_project_school_state_dict = dict(sorted(project_school_state_dict.items(), key=lambda kv: kv[1])) #result a sorted dictionary by number of counts
```

1.3 Text preprocessing

In [0]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [19]:
```
project_data.head(2)
```

Out[19]:

	Unnamed: 0	id	teacher_id	Date	project_title	project_essay_1
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...

In [0]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [21]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
```

I recently read an article about giving students a choice about how they lear n. We already set goals; why not let them choose where to sit, and give them options of what to sit on?I teach at a low-income (Title 1) school. Every yea r, I have a class with a range of abilities, yet they are all the same age. T hey learn differently, and they have different interests. Some have ADHD, and some are fast learners. Yet they are eager and active learners that want and need to be able to move around the room, yet have a place that they can be co mfortable to complete their work.We need a classroom rug that we can use as a class for reading time, and students can use during other learning times. I h ave also requested four Kore Kids wobble chairs and four Back Jack padded por table chairs so that students can still move during whole group lessons witho ut disrupting the class. Having these areas will provide these little ones wi th a way to wiggle while working.Benjamin Franklin once said, \"Tell me and I forget, teach me and I may remember, involve me and I learn.\" I want these c hildren to be involved in their learning by having a choice on where to sit a nd how to learn, all by giving them options for comfortable flexible seating.

==========

At the beginning of every class we start out with a Math Application problem to help students see the relevance of topics in math. We are always in groups and do a lot of cooperative activities. We also use lots of technology in our class. I love seeing my students grow and love math!I have a very diverse pop ulation of students from all different races, SES, and experiences. My studen ts love school and are starting to embrace the hard work it takes to be a fif th grader. My school is a 5th/6th grade school only and is considered a scho ol for the middle grades. It is located in a suburban area. It is now more di verse than it has been in many years.I am in an inclusion setting and many of my students have disabilities. It is hard for them to see the board because o ur resources are old and outdated. A new document camera for our classroom w ill allow our students to see the board more clearly during instructional tim es and will create a classroom environment where lots of movement isn't neces sary just because my students cannot see the board.It's frustrating to teach a lesson when many of my students can't see the board because the resources I have are old and outdated. Oftentimes students will tell me to wait before mo ving on because it takes them forever to write notes because they cannot see the materials. I want students to enjoy coming to my class to learn math and not feel frustrated because they cannot see the board.

==========

My students love coming to school and they love learning. I strive daily to m ake our classroom a relaxed, comfortable and welcoming environment where all learners will excel and grow in their learning. And a new rug will make our d ays even brighter!My 2nd grade classroom is filled with 20 amazing young lear ners. These students fill my heart everyday with their passion for learning n ew things. Working with these students and how engaged they are in each subje ct matter is so much fun. We are small elementary school in mid-Missouri and we have an 80 percent free and reduced lunch rate. I have a wide range of lea rners in my classroom, and all of my students learn in different ways. So it is important to provide a learning environment that meets all students.A beau tiful new carpet will be the focal point of our classroom. The carpet will be full of students all day long. It will be a clean and comfortable place where my students will find comfort in learning. Students will be sitting in small groups, laying and reading a book or even dancing on the carpet for brain bre aks during the day. A carpet in an elementary classroom is the heart of where learning takes place!Thank you for donating or considering a donation to this project. I want to make my 2nd grade classroom as comfortable and inviting as Starbucks or as cozy as a grandma's living room! This beautiful carpet will b e a perfect addition to a classroom the is filled with so much excitement and

```
enthusiasm!
==================================================
```

In [0]:
```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [23]:
```
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

```
I teach in an elementary school that is a 4th / 5th grade building in a small
town in central Illinois. Next year I will be teaching three different classe
s of students reading and language / writing / spelling. In my classroom, my
students enjoy a variety of activities including hands-on and collaborative l
earning in order to help make the information real and interesting to them wh
ile giving them a reason to practice.\r\n\r\nOur students are a wide-variety
of students at our school with over 60% of our students receiving free lunch.
Because of this low-income percentage, our students often require additional
help and support to help make their learning valuable and real-world to them.
Our teachers work hard to collaborate in order to help all of our students ac
hieve at their highest level.\r\n\r\nOur community is very supportive of our
schools, but lately because of lower levels of state support many local busin
esses have cut back on individual assistance for classrooms.  In order to con
tinue some of our learning projects, we have had to look to other support to
help us out.Next year we will be focusing a great deal of our ELA (English La
nguage Arts) time in 5th grade to improving our writing across all the curric
ulum:  math, science, reading, language, and social studies.  These individua
l marker boards will give my students the ability to practice writing skills
individually while giving me the ability to check individual is work as they
practice.  They will also allow them to add creativity to their writing and v
ocabulary practice. \r\n\r\nThese boards are an amazing tool in the classroo
m, and all kids enjoy them! They offer them the benefit of working by themsel
ves and making errors that they can then learn to fix - such an important ste
p in the learning process.nannan
==================================================
```

In [24]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
I teach in an elementary school that is a 4th / 5th grade building in a small
town in central Illinois. Next year I will be teaching three different classe
s of students reading and language / writing / spelling. In my classroom, my
students enjoy a variety of activities including hands-on and collaborative l
earning in order to help make the information real and interesting to them wh
ile giving them a reason to practice.    Our students are a wide-variety of s
tudents at our school with over 60% of our students receiving free lunch. Bec
ause of this low-income percentage, our students often require additional hel
p and support to help make their learning valuable and real-world to them. Ou
r teachers work hard to collaborate in order to help all of our students achi
eve at their highest level.    Our community is very supportive of our school
s, but lately because of lower levels of state support many local businesses
have cut back on individual assistance for classrooms.  In order to continue
some of our learning projects, we have had to look to other support to help u
s out.Next year we will be focusing a great deal of our ELA (English Language
Arts) time in 5th grade to improving our writing across all the curriculum:
math, science, reading, language, and social studies.  These individual marke
r boards will give my students the ability to practice writing skills individ
ually while giving me the ability to check individual is work as they practic
e.  They will also allow them to add creativity to their writing and vocabula
ry practice.     These boards are an amazing tool in the classroom, and all k
ids enjoy them! They offer them the benefit of working by themselves and maki
ng errors that they can then learn to fix - such an important step in the lea
rning process.nannan
```

In [25]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
I teach in an elementary school that is a 4th 5th grade building in a small t
own in central Illinois Next year I will be teaching three different classes
of students reading and language writing spelling In my classroom my students
enjoy a variety of activities including hands on and collaborative learning i
n order to help make the information real and interesting to them while givin
g them a reason to practice Our students are a wide variety of students at ou
r school with over 60 of our students receiving free lunch Because of this lo
w income percentage our students often require additional help and support to
help make their learning valuable and real world to them Our teachers work ha
rd to collaborate in order to help all of our students achieve at their highe
st level Our community is very supportive of our schools but lately because o
f lower levels of state support many local businesses have cut back on indivi
dual assistance for classrooms In order to continue some of our learning proj
ects we have had to look to other support to help us out Next year we will be
focusing a great deal of our ELA English Language Arts time in 5th grade to i
mproving our writing across all the curriculum math science reading language
and social studies These individual marker boards will give my students the a
bility to practice writing skills individually while giving me the ability to
check individual is work as they practice They will also allow them to add cr
eativity to their writing and vocabulary practice These boards are an amazing
tool in the classroom and all kids enjoy them They offer them the benefit of
working by themselves and making errors that they can then learn to fix such
an important step in the learning process nannan
```

In [0]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
, "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he'
, 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "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', 'becau
se', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [27]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:25<00:00, 1974.52it/s]
```

In [28]:
```
# after preprocesing
preprocessed_essays[2000]
```

Out[28]:
```
'i teach elementary school 4th 5th grade building small town central illinois
next year i teaching three different classes students reading language writin
g spelling in classroom students enjoy variety activities including hands col
laborative learning order help make information real interesting giving reaso
n practice our students wide variety students school 60 students receiving fr
ee lunch because low income percentage students often require additional help
support help make learning valuable real world our teachers work hard collabo
rate order help students achieve highest level our community supportive schoo
ls lately lower levels state support many local businesses cut back individua
l assistance classrooms in order continue learning projects look support help
us next year focusing great deal ela english language arts time 5th grade imp
roving writing across curriculum math science reading language social studies
these individual marker boards give students ability practice writing skills
individually giving ability check individual work practice they also allow ad
d creativity writing vocabulary practice these boards amazing tool classroom
kids enjoy they offer benefit working making errors learn fix important step
learning process nannan'
```

In [29]:
```
project_data['preprocessed_essay'] = preprocessed_essays    #create new column
having name project_grad_cat with preprocessed data
project_data.drop(['essay'], axis=1, inplace=True) #delete the project_grade_c
ategory column
project_data.head(2)
```

Out[29]:

	Unnamed: 0	id	teacher_id	Date	project_title	project_essay_1
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	Flexible Seating for Flexible Learning	I recently read an article about giving studen...
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	Going Deep: The Art of Inner Thinking!	My students crave challenge, they eat obstacle...

1.4 Preprocessing of `project_title`

In [0]:
```
# similarly you can preprocess the titles also
```

In [31]:
```
#Printing some random titles
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
```

```
Flexible Seating for Flexible Learning
==================================================
Elmo for Math Instruction
==================================================
There's Only One You in This Great Big World
==================================================
```

In [32]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    title_sent = decontracted(sentance)
    title_sent = title_sent.replace('\\r', ' ')
    title_sent = title_sent.replace('\\"', ' ')
    title_sent = title_sent.replace('\\n', ' ')
    title_sent = re.sub('[^A-Za-z0-9]+', ' ', title_sent)
    # https://gist.github.com/sebleier/554280
    title_sent = ' '.join(e for e in title_sent.split() if e not in stopwords)
    preprocessed_titles.append(title_sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:01<00:00, 40746.58it/s]
```

In [33]:
```
project_data['preprocessed_title'] = preprocessed_titles    #create new column having name project_grad_cat with preprocessed data
project_data.drop(['project_title'], axis=1, inplace=True) #delete the project_grade_category column
project_data.head()
```

Out[33]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_ess
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a income (Ti school. E
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an u publi eleme scho
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My stu d challe movement
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Lang Arts and S Justice Ma
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I hav privile teachi incredi

In [34]:
```
project_data.head(5)
```

Out[34]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_ess
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a income (Ti school. E
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an u publi eleme scho
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My stu d challe movement
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Lang Arts and S Justice Ma
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I hav privile teachi incredi

1.5 Preparing data for models

In [35]:
```
project_data.columns
```

Out[35]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'Date', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'teach_pref', 'project_grad_cat', 'project_school_state',
       'preprocessed_essay', 'preprocessed_title'],
      dtype='object')
```

In [0]:
```
def count(line):
    num_text=[]
    for words in line:
        splitted = words.split()
        length = len(splitted)
        num_text.append(length)
    return num_text
```

In [35]:
```
project_data['num_title'] = count(project_data['preprocessed_title'])    #crea
te new column having name project_grad_cat with preprocessed data
project_data.head(5)
```

Out[35]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_ess
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a income (Ti school. E
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an u publi eleme scho
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My stu d challe movement
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Lang Arts and S Justice Ma
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I hav privile teachi incredi

5 rows × 21 columns

In [36]:
```
project_data['num_essay'] = count(project_data['preprocessed_essay'])    #crea
te new column having name project_grad_cat with preprocessed data
project_data.head(5)
```

Out[36]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_ess
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a income (Ti school. E
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an u publi eleme scho
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My stu d challe movement
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Lang Arts and S Justice Ma
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I hav privile teachi incredi

5 rows × 22 columns

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [0]:
```
# you can do the similar thing with state, teacher_prefix and project_grade_ca
tegory also
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [0]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

1.5.2.3 Using Pretrained Models: Avg W2V

In [39]:
```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)
```

```
'''
```

Out[39]:
```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/40
84039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n
f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in t
qdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        model[wo
rd] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return
model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ================
==========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/
s]\nDone. 1917495  words loaded!\n\n# ============================\n\nwords =
[]\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in p
reproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in t
he coupus", len(words))\nwords = set(words)\nprint("the unique words in the c
oupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\np
rint("The number of words that are present in both glove vectors and our coup
us",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,
3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in wor
ds:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("wo
rd 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle
files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-v
ariables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as
f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [0]:
```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

1.5.4 Merging all the above features

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

Computing Sentiment Scores

In [42]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.downloader.download('vader_lexicon')

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the
smallest students with the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses
 and multiple intelligences i use a wide range\
of techniques to help all my students succeed students in my class come from a
variety of different backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans o
ur school is a caring community of successful \
learners which can be seen through collaborative student project based learnin
g in and out of the classroom kindergarteners \
in my class love to work with hands on materials and have many different oppor
tunities to practice a skill before it is\
mastered having the social skills to work cooperatively with friends is a cruc
ial aspect of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my stude
nts love to role play in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try coo
king with real food i will take their idea \
and create common core cooking lessons where we learn important math and writi
ng concepts while cooking delicious healthy \
food for snack time my students will have a grounded appreciation for the work
that went into making the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodi
es this project would expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to
make homemade applesauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also
create our own cookbooks to be printed and \
shared with families students will gain math and literature skills as well as
 a life long enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

In [43]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.downloader.download('vader_lexicon')

# import nltk
# nltk.download('vader_lexicon')
#https://www.programcreek.com/python/example/100005/nltk.sentiment.vader.SentimentIntensityAnalyzer

def analyze_sentiment(project_data):
    sentiments = []
    sid = SentimentIntensityAnalyzer()
    for i in range(project_data.shape[0]):
        line = project_data['preprocessed_essay'].iloc[i]
        sentiment = sid.polarity_scores(line)
        sentiments.append([sentiment['neg'], sentiment['pos'],
                           sentiment['neu'], sentiment['compound']])
    project_data[['neg', 'pos', 'neu', 'compound']] = pd.DataFrame(sentiments)
    return project_data
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [44]:
```
project_data=analyze_sentiment(project_data)
project_data.head(5)
```

Out[44]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_ess
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a income (Ti school. E
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an u publi eleme scho
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My stu d challe movement
3	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Lang Arts and S Justice Ma
4	57854	p099430	4000cfe0c8b2df75a218347c1765e283	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I hav privile teachi incredi

5 rows × 26 columns

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-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. knn

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

In [0]:
```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debug
ging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the rea
der
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [46]:
```
y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
project_data.head(3)
```

Out[46]:

	Unnamed: 0	id	teacher_id	Date	project_essay_1	project_ess
0	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a income (Ti school. E
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an u publi eleme scho
2	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My stu d challe movement

3 rows × 25 columns

In [0]:
```
#train_test_split
from sklearn.model_selection import train_test_split
project_data_train, project_data_test, project_data_y_train, project_data_y_te
st = train_test_split(project_data, y, test_size=0.33, stratify=y)
project_data_train, project_data_cv, project_data_y_train, project_data_y_cv =
train_test_split(project_data_train, project_data_y_train, test_size=0.33, str
atify=project_data_y_train)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [0]:
```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debug
ging your code
# make sure you featurize train and test data separatly

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

In [49]:
```
#Teacher Prefix
#https://stackoverflow.com/questions/48090658/sklearn-how-to-incorporate-missing-data-when-one-hot-encoding
from sklearn.feature_extraction.text import CountVectorizer
vectorizer3 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer3.fit(project_data_train['clean_categories'].values)
print(vectorizer3.get_feature_names())

categories_one_hot_train = vectorizer3.transform(project_data_train['clean_categories'].values)
categories_one_hot_cv = vectorizer3.transform(project_data_cv['clean_categories'].values)
categories_one_hot_test = vectorizer3.transform(project_data_test['clean_categories'].values)

print("After vectorizations")
print(categories_one_hot_train.shape, project_data_y_train.shape)
print(categories_one_hot_cv.shape, project_data_y_cv.shape)
print(categories_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

In [50]:
```
#Teacher Prefix
#https://stackoverflow.com/questions/48090658/sklearn-how-to-incorporate-missing-data-when-one-hot-encoding
from sklearn.feature_extraction.text import CountVectorizer
vectorizer4 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer4.fit(project_data_train['clean_subcategories'].values)
print(vectorizer4.get_feature_names())

sub_categories_one_hot_train = vectorizer4.transform(project_data_train['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer4.transform(project_data_cv['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer4.transform(project_data_test['clean_subcategories'].values)

print("After vectorizations")
print(sub_categories_one_hot_train.shape, project_data_y_train.shape)
print(sub_categories_one_hot_cv.shape, project_data_y_cv.shape)
print(sub_categories_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

In [51]:
```
#Teacher Prefix
#https://stackoverflow.com/questions/48090658/sklearn-how-to-incorporate-missing-data-when-one-hot-encoding
from sklearn.feature_extraction.text import CountVectorizer
vectorizer1 = CountVectorizer(vocabulary=list(sorted_teach_pref_dict.keys()), lowercase=False, binary=True)
vectorizer1.fit(project_data_train['teach_pref'].values)
print(vectorizer1.get_feature_names())

teach_pref_one_hot_train = vectorizer1.transform(project_data_train['teach_pref'].values)
teach_pref_one_hot_cv = vectorizer1.transform(project_data_cv['teach_pref'].values)
teach_pref_one_hot_test = vectorizer1.transform(project_data_test['teach_pref'].values)

print("After vectorizations")
print(teach_pref_one_hot_train.shape, project_data_y_train.shape)
print(teach_pref_one_hot_cv.shape, project_data_y_cv.shape)
print(teach_pref_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

```
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
After vectorizations
(22445, 5) (22445,)
(11055, 5) (11055,)
(16500, 5) (16500,)
====================================================================================================
```

In [52]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grad_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(project_data['project_grad_cat'].values)
print(vectorizer.get_feature_names())

project_grad_one_hot_train = vectorizer.transform(project_data_train['project_grad_cat'].values)  #this will change categorical data into binary form
project_grad_one_hot_cv = vectorizer.transform(project_data_cv['project_grad_cat'].values)  #this will change categorical data into binary form
project_grad_one_hot_test = vectorizer.transform(project_data_test['project_grad_cat'].values)  #this will change categorical data into binary form

print("After vectorizations")
print(project_grad_one_hot_train.shape, project_data_y_train.shape)
print(project_grad_one_hot_cv.shape, project_data_y_cv.shape)
print(project_grad_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

```
['Grades9-12', 'Grades6-8', 'Grades3-5', 'GradesPreK-2']
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
====================================================================================================
```

In [53]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer2 = CountVectorizer(vocabulary=list(sorted_project_school_state_dict
.keys()), lowercase=False, binary=True)
vectorizer2.fit(project_data['project_school_state'].values)
print(vectorizer2.get_feature_names())

project_school_state_one_hot_train = vectorizer2.transform(project_data_train[
'project_school_state'].values)  #this will change categorical data into binar
y form
project_school_state_one_hot_cv = vectorizer2.transform(project_data_cv['proje
ct_school_state'].values)  #this will change categorical data into binary form
project_school_state_one_hot_test = vectorizer2.transform(project_data_test['p
roject_school_state'].values)  #this will change categorical data into binary
 form

print("After vectorizations")
print(project_school_state_one_hot_train.shape, project_data_y_train.shape)
print(project_school_state_one_hot_cv.shape, project_data_y_cv.shape)
print(project_school_state_one_hot_test.shape, project_data_y_test.shape)
print("="*100)
```

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

Encoding numerical features

In [54]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['quantity'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
num_title_standardized_train = num_title_scalar.transform(project_data_train['num_title'].values.reshape(-1, 1))
num_title_standardized_cv = num_title_scalar.transform(project_data_cv['num_title'].values.reshape(-1, 1))
num_title_standardized_test = num_title_scalar.transform(project_data_test['num_title'].values.reshape(-1, 1))
```

```
Mean : 4.32706616172867, Standard deviation : 1.776784113754445

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

In [55]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['quantity'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
num_essay_standardized_train = num_title_scalar.transform(project_data_train['num_essay'].values.reshape(-1, 1))
num_essay_standardized_cv = num_title_scalar.transform(project_data_cv['num_essay'].values.reshape(-1, 1))
num_essay_standardized_test = num_title_scalar.transform(project_data_test['num_essay'].values.reshape(-1, 1))
```

```
Mean : 151.58966362218757, Standard deviation : 39.25640578935296

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

In [56]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['quantity'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
quantity_standardized_train = quantity_scalar.transform(project_data_train['quantity'].values.reshape(-1, 1))
quantity_standardized_cv = quantity_scalar.transform(project_data_cv['quantity'].values.reshape(-1, 1))
quantity_standardized_test = quantity_scalar.transform(project_data_test['quantity'].values.reshape(-1, 1))
```

```
Mean : 18.63558205105633, Standard deviation : 7.343398311429823
```

In [57]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(project_data_train['price'].values.reshape(-1, 1))
price_standardized_cv = price_scalar.transform(project_data_cv['price'].values.reshape(-1, 1))
price_standardized_test = price_scalar.transform(project_data_test['price'].values.reshape(-1, 1))
```

```
Mean : 288.49892259123834, Standard deviation : 72.9090255358347
```

In [58]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['teacher_number_of_previously_posted_projects
#'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
previous_standardized_train = previous_scalar.transform(project_data_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
previous_standardized_cv = previous_scalar.transform(project_data_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
previous_standardized_test = previous_scalar.transform(project_data_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
```

```
Mean : 11.10202717754511, Standard deviation : 27.62833104297554

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:595: DataConversionWarning:

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

In [59]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['teacher_number_of_previously_posted_projects
#'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
neg_standardized_train = neg_scalar.transform(project_data_train['neg'].values.reshape(-1, 1))
neg_standardized_cv = neg_scalar.transform(project_data_cv['neg'].values.reshape(-1, 1))
neg_standardized_test = neg_scalar.transform(project_data_test['neg'].values.reshape(-1, 1))
```

```
Mean : 0.04524027623078638, Standard deviation : 0.03413733691868242
```

In [60]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['teacher_number_of_previously_posted_projects
#'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
pos_standardized_train = pos_scalar.transform(project_data_train['pos'].values.reshape(-1, 1))
pos_standardized_cv = pos_scalar.transform(project_data_cv['pos'].values.reshape(-1, 1))
pos_standardized_test = pos_scalar.transform(project_data_test['pos'].values.reshape(-1, 1))
```

```
Mean : 0.26760467810202715, Standard deviation : 0.07415542530520912
```

In [61]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['teacher_number_of_previously_posted_projects
#'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
neu_standardized_train = neu_scalar.transform(project_data_train['neu'].values.reshape(-1, 1))
neu_standardized_cv = neu_scalar.transform(project_data_cv['neu'].values.reshape(-1, 1))
neu_standardized_test = neu_scalar.transform(project_data_test['neu'].values.reshape(-1, 1))
```

```
Mean : 0.04524027623078638, Standard deviation : 0.03413733691868242
```

In [62]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# quantity_standardized = standardScalar.fit(project_data['teacher_number_of_previously_posted_projects
#'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
compound_standardized_train = compound_scalar.transform(project_data_train['compound'].values.reshape(-1, 1))
compound_standardized_cv = compound_scalar.transform(project_data_cv['compound'].values.reshape(-1, 1))
compound_standardized_test = compound_scalar.transform(project_data_test['compound'].values.reshape(-1, 1))
```

```
Mean : 0.04524027623078638, Standard deviation : 0.03413733691868242
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [0]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

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

Bow on title and essay

In [64]:
```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
vectorizer_bow = CountVectorizer(min_df=10)
vectorizer_bow = vectorizer_bow.fit(project_data_train['preprocessed_title'])
#this will convert word into n dimensional vectors

title_bow_train = vectorizer_bow.transform(project_data_train['preprocessed_title'].values)
title_bow_cv = vectorizer_bow.transform(project_data_cv['preprocessed_title'].values)
title_bow_test = vectorizer_bow.transform(project_data_test['preprocessed_title'].values)

print("Shape of train matrix after one hot encodig ",title_bow_train.shape)
print("Shape of cv matrix after one hot encodig ",title_bow_cv.shape)
print("Shape of test matrix after one hot encodig ",title_bow_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 1236)
Shape of cv matrix after one hot encodig  (11055, 1236)
Shape of test matrix after one hot encodig  (16500, 1236)
```

In [65]:
```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
vectorizer_bow1 = CountVectorizer(min_df=10, ngram_range=(1,2), max_features=5000)
vectorizer_bow1 = vectorizer_bow1.fit(project_data_train['preprocessed_essay']) #this will convert word into n dimensional vectors

essay_bow_train = vectorizer_bow1.transform(project_data_train['preprocessed_essay'].values)
essay_bow_cv = vectorizer_bow1.transform(project_data_cv['preprocessed_essay'].values)
essay_bow_test = vectorizer_bow1.transform(project_data_test['preprocessed_essay'].values)

print("Shape of train matrix after one hot encodig ",essay_bow_train.shape)
print("Shape of cv matrix after one hot encodig ",essay_bow_cv.shape)
print("Shape of test matrix after one hot encodig ",essay_bow_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 5000)
Shape of cv matrix after one hot encodig  (11055, 5000)
Shape of test matrix after one hot encodig  (16500, 5000)
```

tdidf on essay and title

In [66]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf = TfidfVectorizer(min_df=10)
vectorizer_tfidf = vectorizer_tfidf.fit(project_data_train['preprocessed_titl
e'])

title_tfidf_train = vectorizer_tfidf.transform(project_data_train['preprocesse
d_title'].values)
title_tfidf_cv = vectorizer_tfidf.transform(project_data_cv['preprocessed_titl
e'].values)
title_tfidf_test = vectorizer_tfidf.transform(project_data_test['preprocessed_
title'].values)

print("Shape of train matrix after one hot encodig ",title_tfidf_train.shape)
print("Shape of cv matrix after one hot encodig ",title_tfidf_cv.shape)
print("Shape of test matrix after one hot encodig ",title_tfidf_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 1236)
Shape of cv matrix after one hot encodig  (11055, 1236)
Shape of test matrix after one hot encodig  (16500, 1236)
```

In [67]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf1 = TfidfVectorizer(min_df=10, ngram_range=(1,2), max_features
=5000)
vectorizer_tfidf1 = vectorizer_tfidf1.fit(project_data_train['preprocessed_ess
ay'])


essay_tfidf_train = vectorizer_tfidf1.transform(project_data_train['preprocess
ed_essay'].values)
essay_tfidf_cv = vectorizer_tfidf1.transform(project_data_cv['preprocessed_ess
ay'].values)
essay_tfidf_test = vectorizer_tfidf1.transform(project_data_test['preprocessed
_essay'].values)

print("Shape of train matrix after one hot encodig ",essay_bow_train.shape)
print("Shape of cv matrix after one hot encodig ",essay_bow_cv.shape)
print("Shape of test matrix after one hot encodig ",essay_bow_test.shape)
```

```
Shape of train matrix after one hot encodig  (22445, 5000)
Shape of cv matrix after one hot encodig  (11055, 5000)
Shape of test matrix after one hot encodig  (16500, 5000)
```

Using pretrained glove vectors

In [0]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how
-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('/content/drive/My Drive/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [69]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored i
n this list
for sentence in tqdm(project_data_train['preprocessed_essay']): # for each rev
iew/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

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

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in t
his list
for sentence in tqdm(project_data_cv['preprocessed_essay']): # for each revie
w/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(project_data_test['preprocessed_essay']): # for each revi
ew/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
100%|██████████| 22445/22445 [00:06<00:00, 3510.92it/s]
  3%|▌          | 341/11055 [00:00<00:03, 3403.36it/s]

22445
300

100%|██████████| 11055/11055 [00:02<00:00, 3752.29it/s]
  2%|▏          | 346/16500 [00:00<00:04, 3454.76it/s]

11055
300

100%|██████████| 16500/16500 [00:04<00:00, 3718.02it/s]

16500
300
```

In [70]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train1 = []; # the avg-w2v for each sentence/review is stored
 in this list
for sentence in tqdm(project_data_train['preprocessed_title']): # for each rev
iew/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train1.append(vector)

print(len(avg_w2v_vectors_train1))
print(len(avg_w2v_vectors_train1[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_cv1 = []; # the avg-w2v for each sentence/review is stored in
 this list
for sentence in tqdm(project_data_cv['preprocessed_title']): # for each revie
w/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv1.append(vector)

print(len(avg_w2v_vectors_cv1))
print(len(avg_w2v_vectors_cv1[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_test1 = []; # the avg-w2v for each sentence/review is stored i
n this list
for sentence in tqdm(project_data_test['preprocessed_title']): # for each revi
ew/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test1.append(vector)

print(len(avg_w2v_vectors_test1))
print(len(avg_w2v_vectors_test1[0]))
```

```
100%|██████████| 22445/22445 [00:00<00:00, 66000.42it/s]
100%|██████████| 11055/11055 [00:00<00:00, 66464.69it/s]
  0%|          | 0/16500 [00:00<?, ?it/s]

22445
300
11055
300

100%|██████████| 16500/16500 [00:00<00:00, 64071.87it/s]

16500
300
```

In [71]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored
in this list
for sentence in tqdm(project_data_train['preprocessed_essay']): # for each rev
iew/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/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((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
    tfidf_w2v_vectors_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in
this list
for sentence in tqdm(project_data_cv['preprocessed_essay']): # for each revie
w/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/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((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
    tfidf_w2v_vectors_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
```

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

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test= []; # the avg-w2v for each sentence/review is stored i
n this list
for sentence in tqdm(project_data_test['preprocessed_essay']): # for each revi
ew/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/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((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
    tfidf_w2v_vectors_test.append(vector)

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

```
100%|██████████| 22445/22445 [00:38<00:00, 577.68it/s]
  1%|          | 59/11055 [00:00<00:18, 582.99it/s]

22445
300

100%|██████████| 11055/11055 [00:19<00:00, 568.09it/s]
  0%|          | 49/16500 [00:00<00:33, 486.39it/s]

11055
300

100%|██████████| 16500/16500 [00:29<00:00, 563.21it/s]

16500
300
```

In [72]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model_1 = TfidfVectorizer()
tfidf_model_1.fit(project_data_train['preprocessed_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_1.get_feature_names(), list(tfidf_model_1.id
f_)))
tfidf_words = set(tfidf_model_1.get_feature_names())

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train1 = []; # the avg-w2v for each sentence/review is store
d in this list
for sentence in tqdm(project_data_train['preprocessed_title']): # for each rev
iew/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/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((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
    tfidf_w2v_vectors_train1.append(vector)

print(len(tfidf_w2v_vectors_train1))
print(len(tfidf_w2v_vectors_train1[0]))

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv1 = []; # the avg-w2v for each sentence/review is stored i
n this list
for sentence in tqdm(project_data_cv['preprocessed_title']): # for each revie
w/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/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((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
    tfidf_w2v_vectors_cv1.append(vector)

print(len(tfidf_w2v_vectors_cv1))
```

```
print(len(tfidf_w2v_vectors_cv1[0]))

# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test1 = []; # the avg-w2v for each sentence/review is stored
in this list
for sentence in tqdm(project_data_test['preprocessed_title']): # for each revi
ew/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/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((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
    tfidf_w2v_vectors_test1.append(vector)

print(len(tfidf_w2v_vectors_test1))
print(len(tfidf_w2v_vectors_test1[0]))
```

```
100%|██████████| 22445/22445 [00:00<00:00, 27594.19it/s]
 28%|██▊       | 3114/11055 [00:00<00:00, 31128.45it/s]

22445
300

100%|██████████| 11055/11055 [00:00<00:00, 33582.28it/s]
 14%|█▍        | 2255/16500 [00:00<00:00, 22547.71it/s]

11055
300

100%|██████████| 16500/16500 [00:00<00:00, 26035.33it/s]

16500
300
```

In [0]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
train_1=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, title_bow_train, essay_bow_train)).tocsr()
cv_1=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, title_bow_cv, essay_bow_cv)).tocsr()
test_1=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, title_bow_test,essay_bow_test)).tocsr()
```

In [0]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
train_2=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, title_tfidf_train, essay_tfidf_train)).tocsr()
cv_2=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, title_tfidf_cv, essay_tfidf_cv)).tocsr()
test_2=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, title_tfidf_test,essay_tfidf_test)).tocsr()
```

In [0]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
train_3=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, avg_w2v_vectors_train1, avg_w2v_vectors_train)).tocsr()
cv_3=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, avg_w2v_vectors_cv1, avg_w2v_vectors_cv)).tocsr()
test_3=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, avg_w2v_vectors_test1,avg_w2v_vectors_test)).tocsr()
```

In [0]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
train_4=hstack((teach_pref_one_hot_train, categories_one_hot_train, sub_categories_one_hot_train, project_grad_one_hot_train, price_standardized_train, quantity_standardized_train, previous_standardized_train, tfidf_w2v_vectors_train1, tfidf_w2v_vectors_train)).tocsr()
cv_4=hstack((teach_pref_one_hot_cv, categories_one_hot_cv, sub_categories_one_hot_cv, project_grad_one_hot_cv, price_standardized_cv, quantity_standardized_cv, previous_standardized_cv, tfidf_w2v_vectors_cv1, tfidf_w2v_vectors_cv)).tocsr()
test_4=hstack((teach_pref_one_hot_test, categories_one_hot_test, sub_categories_one_hot_test, project_grad_one_hot_test, price_standardized_test, quantity_standardized_test, previous_standardized_test, tfidf_w2v_vectors_test1,tfidf_w2v_vectors_test)).tocsr()
```

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 instrucations

In [0]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Reference source: AppliedAi course Facebook recommendation casestudy
https://stackoverflow.com/questions/37902459/seaborn-color-palette-as-matplotlib-colormap (https://stackoverflow.com/questions/37902459/seaborn-color-palette-as-matplotlib-colormap) https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html (https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)
https://seaborn.pydata.org/generated/seaborn.heatmap.html (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

Numeric+Categoric+BOW

In [0]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates 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 will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

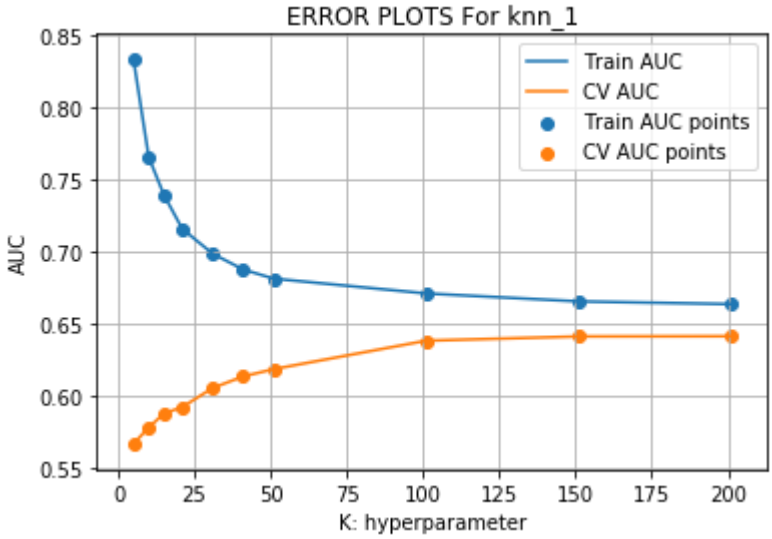
In [ ]:
```
# Below codes are taken from applied ai course facebook friend recommendation
#https://www.appliedaicourse.com/
```

In [0]:
```
from sklearn.metrics import confusion_matrix
def plot_confusion_matrix(test_y, predict_y):
    C = confusion_matrix(test_y, predict_y)

    A =(((C.T)/(C.sum(axis=1))).T)

    B =(C/C.sum(axis=0))
    plt.figure(figsize=(20,4))

    labels = [0,1]
    # representing A in heatmap format
    cmap=sns.light_palette("Navy", as_cmap=True)#https://stackoverflow.com/que
stions/37902459/seaborn-color-palette-as-matplotlib-colormap
    plt.subplot(1, 3, 1)
    sns.heatmap(C, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, ytick
labels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Confusion matrix")

    plt.subplot(1, 3, 2)
    sns.heatmap(B, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, ytick
labels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Precision matrix")

    plt.subplot(1, 3, 3)
    # representing B in heatmap format
    sns.heatmap(A, annot=True, cmap=cmap, fmt=".3f", xticklabels=labels, ytick
labels=labels)
    plt.xlabel('Predicted Class')
    plt.ylabel('Original Class')
    plt.title("Recall matrix")
    plt.show()
```

In [0]:
```
def Zero1(prediction):
    predicted=[]
    for i in prediction:
        if i<0.5:
            predicted.append(0)
        else:
            predicted.append(1)
    return predicted
```

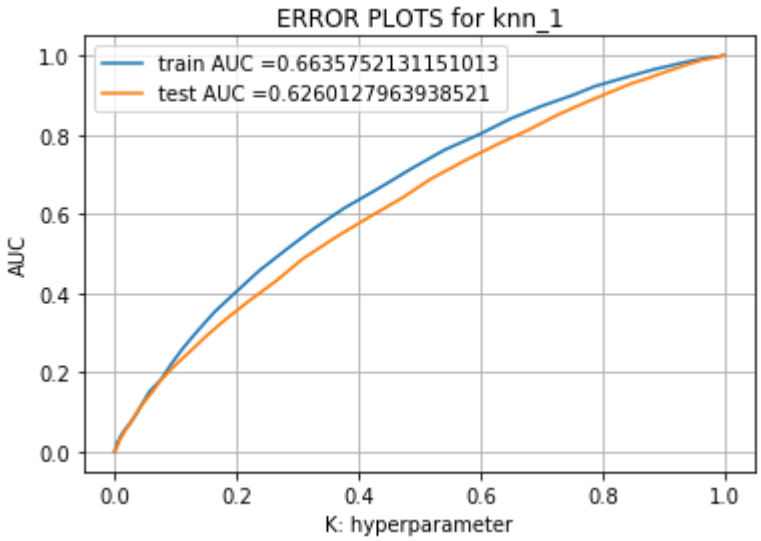
In [82]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

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

"""

train_auc = []
cv_auc = []
K = [5, 10, 15, 21, 31, 41, 51, 101, 151, 201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(train_1, project_data_y_train)

    y_train_pred = batch_predict(neigh, train_1)
    y_cv_pred = batch_predict(neigh, cv_1)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(project_data_y_train,y_train_pred))
    cv_auc.append(roc_auc_score(project_data_y_cv, y_cv_pred))

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

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

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS For knn_1")
plt.grid()
plt.show()
```

```
100%|██████████| 10/10 [17:02<00:00, 102.31s/it]
```



In [0]:
```
# from the error plot we choose K such that, we will have maximum AUC on cv da
ta and gap between the train and cv is less
# Note: based on the method you use you might get different hyperparameter val
ues as best one
# so, we choose according to the method you choose, you use gridsearch if you
 are having more computing power and note it will take more time
# if we increase the cv values in the GridSearchCV you will get more rebust re
sults.

#here we are choosing the best_k based on forloop results
best_k = 181
```

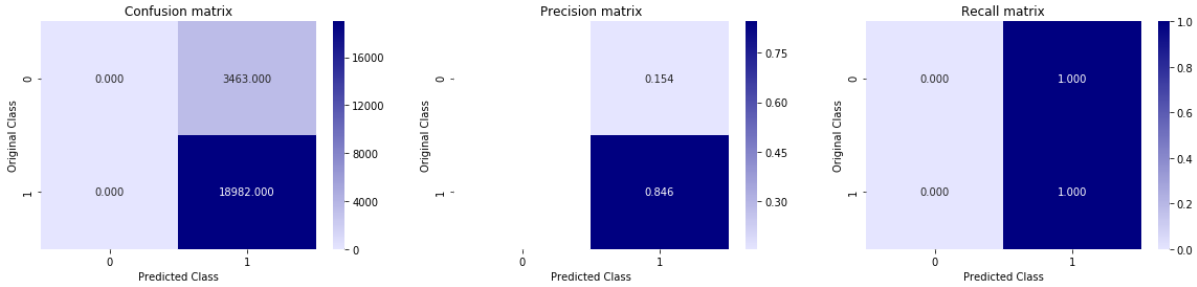
In [84]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k, algorithm='brute')
neigh.fit(train_1, project_data_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, train_1)
y_test_pred = batch_predict(neigh, test_1)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS for knn_1")
plt.grid()
plt.show()
```



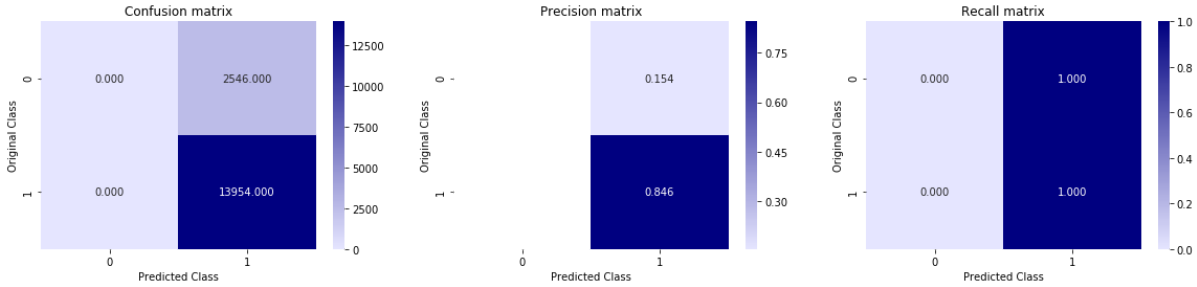
Observation: we can see that train accuracy =66.35% and test accuracy =62.60%.

In [85]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```




```
Test confusion_matrix
```

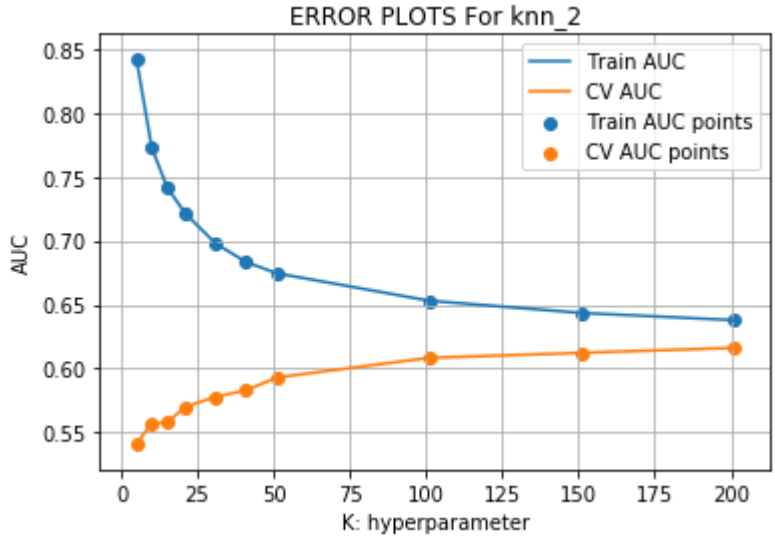


In [0]:

Numeric+Categoric+tfidf

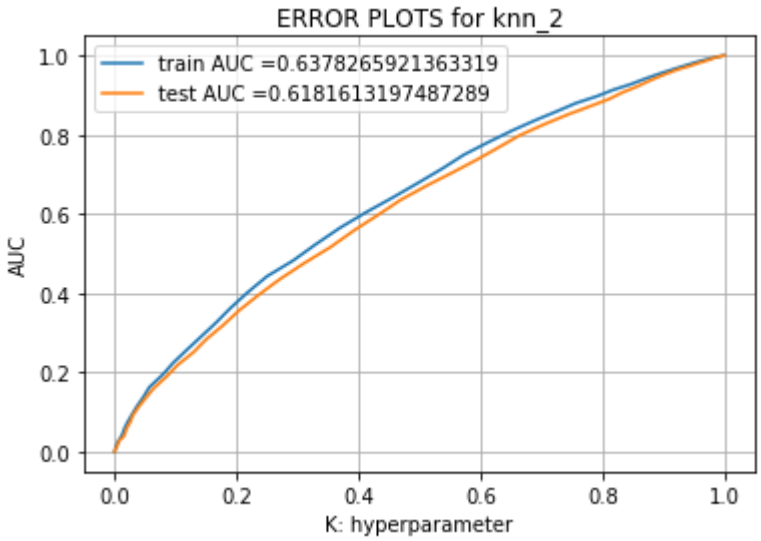
In [86]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

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

"""

train_auc_2 = []
cv_auc_2 = []
K = [5, 10, 15, 21, 31, 41, 51, 101, 151, 201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(train_2, project_data_y_train)

    y_train_pred = batch_predict(neigh, train_2)
    y_cv_pred = batch_predict(neigh, cv_2)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs
    train_auc_2.append(roc_auc_score(project_data_y_train,y_train_pred))
    cv_auc_2.append(roc_auc_score(project_data_y_cv, y_cv_pred))

plt.plot(K, train_auc_2, label='Train AUC')
plt.plot(K, cv_auc_2, label='CV AUC')

plt.scatter(K, train_auc_2, label='Train AUC points')
plt.scatter(K, cv_auc_2, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS For knn_2")
plt.grid()
plt.show()
```

```
100%|██████████| 10/10 [17:02<00:00, 102.66s/it]
```



In [0]:
```
# from the error plot we choose K such that, we will have maximum AUC on cv da
ta and gap between the train and cv is less
# Note: based on the method you use you might get different hyperparameter val
ues as best one
# so, we choose according to the method you choose, you use gridsearch if you
 are having more computing power and note it will take more time
# if we increase the cv values in the GridSearchCV you will get more rebust re
sults.

#here we are choosing the best_k based on forloop results
best_k = 201
```

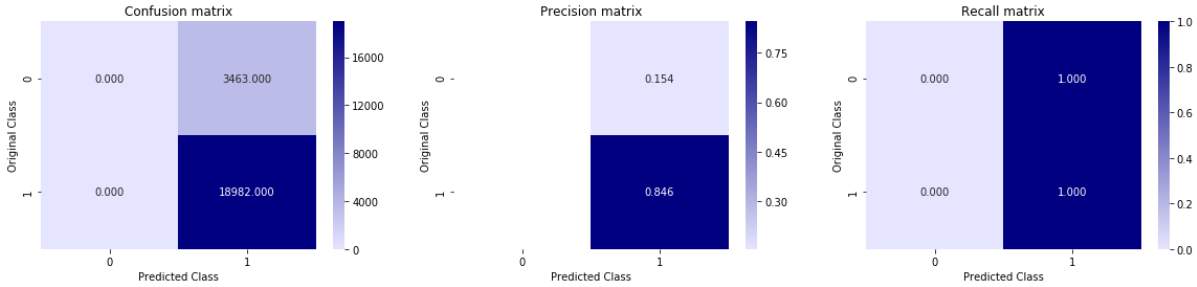
In [88]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k, algorithm='brute')
neigh.fit(train_2, project_data_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, train_2)
y_test_pred = batch_predict(neigh, test_2)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS for knn_2")
plt.grid()
plt.show()
```



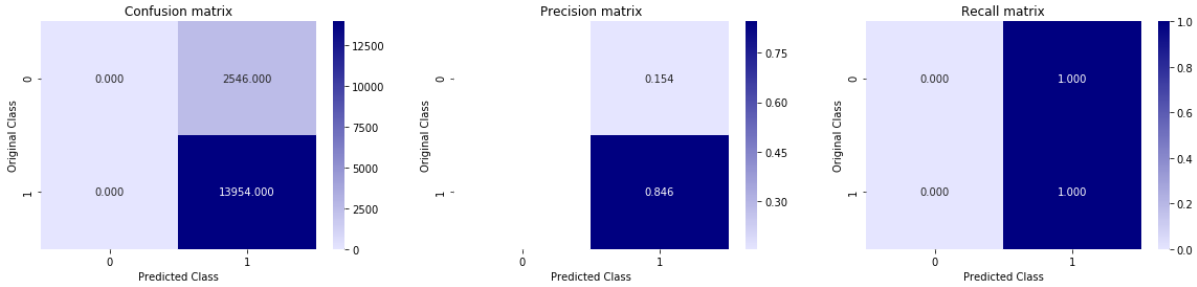
Observation: train auc=63.78%, test auc=61.85%

In [89]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```



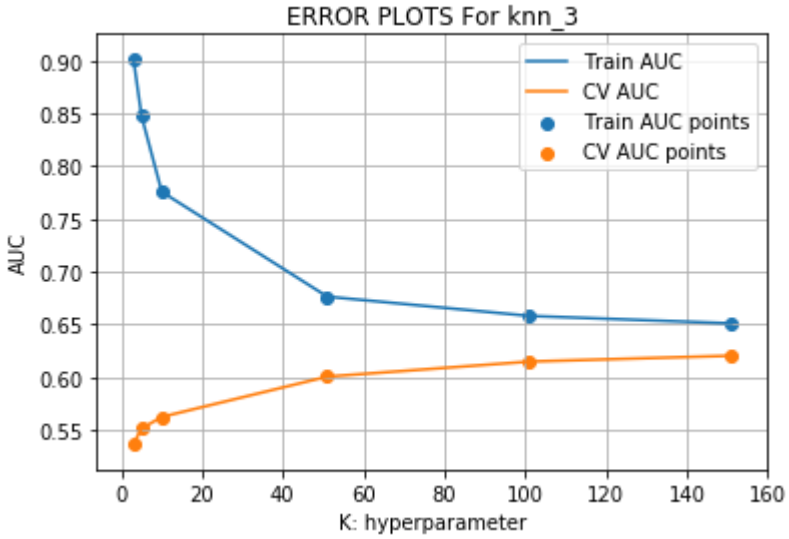

```
Test confusion_matrix
```



Numeric+Categoric+avgw2v

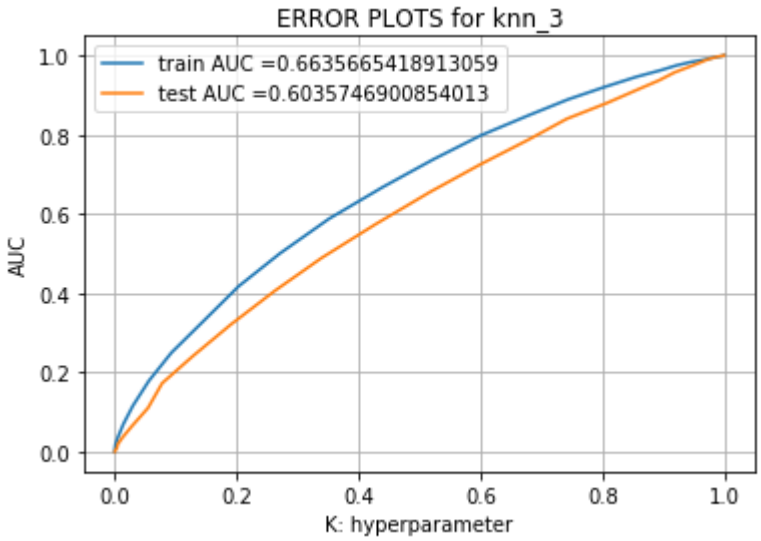
In [90]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
from sklearn.neighbors import NearestNeighbors
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

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

"""

train_auc_3 = []
cv_auc_3 = []
K = [3, 5, 10, 51, 101, 151]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm="brute")
    neigh.fit(train_3, project_data_y_train)

    y_train_pred = batch_predict(neigh, train_3)
    y_cv_pred = batch_predict(neigh, cv_3)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs
    train_auc_3.append(roc_auc_score(project_data_y_train,y_train_pred))
    cv_auc_3.append(roc_auc_score(project_data_y_cv, y_cv_pred))

plt.plot(K, train_auc_3, label='Train AUC')
plt.plot(K, cv_auc_3, label='CV AUC')

plt.scatter(K, train_auc_3, label='Train AUC points')
plt.scatter(K, cv_auc_3, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS For knn_3")
plt.grid()
plt.show()
```

```
100%|██████████| 6/6 [3:13:41<00:00, 1938.85s/it]
```



In [0]:
```
# from the error plot we choose K such that, we will have maximum AUC on cv da
ta and gap between the train and cv is less
# Note: based on the method you use you might get different hyperparameter val
ues as best one
# so, we choose according to the method you choose, you use gridsearch if you
 are having more computing power and note it will take more time
# if we increase the cv values in the GridSearchCV you will get more rebust re
sults.

#here we are choosing the best_k based on forloop results
best_k = 81
```

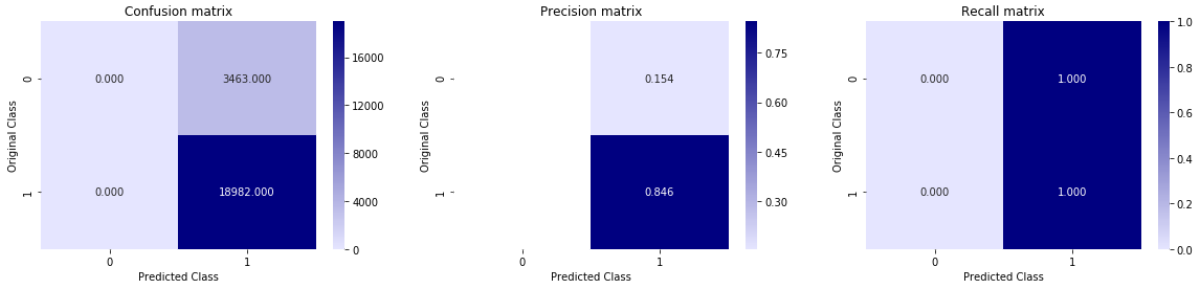
In [92]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k, algorithm='brute')
neigh.fit(train_3, project_data_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, train_3)
y_test_pred = batch_predict(neigh, test_3)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS for knn_3")
plt.grid()
plt.show()
```



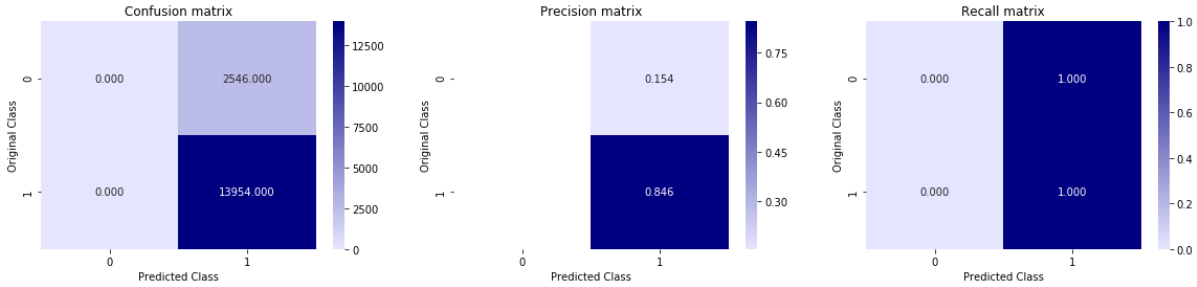
Observation: train acc= 66.35%, test auc=60.35%

In [93]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```




```
Test confusion_matrix
```

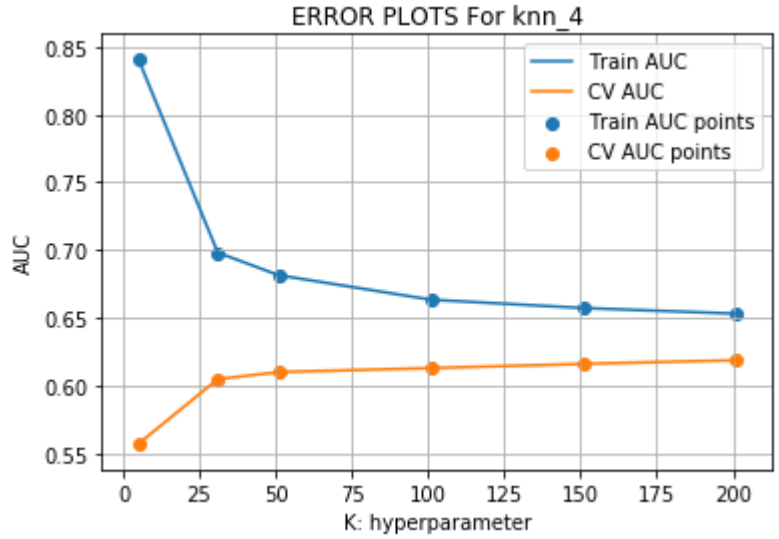


In [0]:

Numeric+Categoric+tfidfw2v

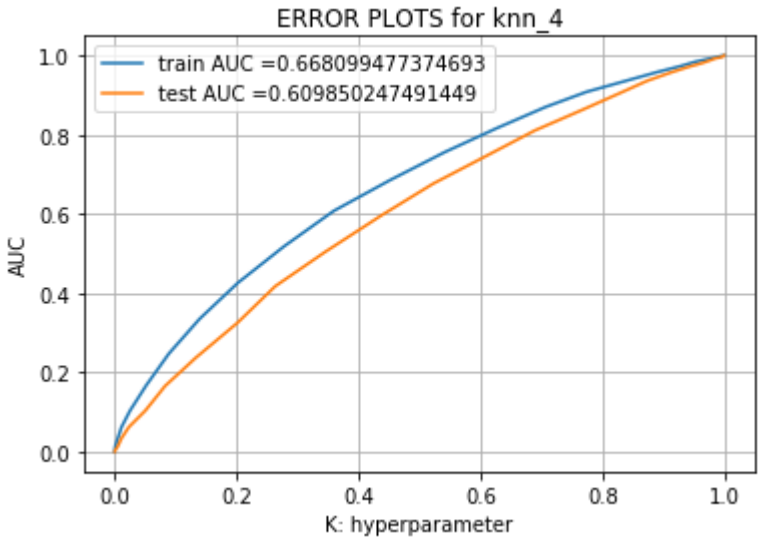
In [87]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

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

"""

train_auc_4 = []
cv_auc_4 = []
K = [5, 31, 51, 101, 151, 201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(train_4, project_data_y_train)

    y_train_pred = batch_predict(neigh, train_4)
    y_cv_pred = batch_predict(neigh, cv_4)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs
    train_auc_4.append(roc_auc_score(project_data_y_train,y_train_pred))
    cv_auc_4.append(roc_auc_score(project_data_y_cv, y_cv_pred))

plt.plot(K, train_auc_4, label='Train AUC')
plt.plot(K, cv_auc_4, label='CV AUC')

plt.scatter(K, train_auc_4, label='Train AUC points')
plt.scatter(K, cv_auc_4, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS For knn_4")
plt.grid()
plt.show()
```

```
100%|██████████| 6/6 [3:20:35<00:00, 2003.73s/it]
```



In [0]:
```
# from the error plot we choose K such that, we will have maximum AUC on cv da
ta and gap between the train and cv is less
# Note: based on the method you use you might get different hyperparameter val
ues as best one
# so, we choose according to the method you choose, you use gridsearch if you
 are having more computing power and note it will take more time
# if we increase the cv values in the GridSearchCV you will get more rebust re
sults.

#here we are choosing the best_k based on forloop results
best_k = 71
```

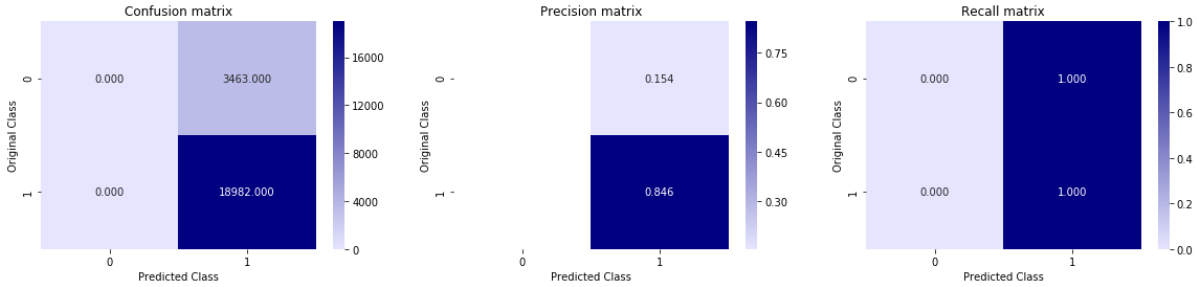
In [89]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k)
neigh.fit(train_4, project_data_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, train_4)
y_test_pred = batch_predict(neigh, test_4)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS for knn_4")
plt.grid()
plt.show()
```



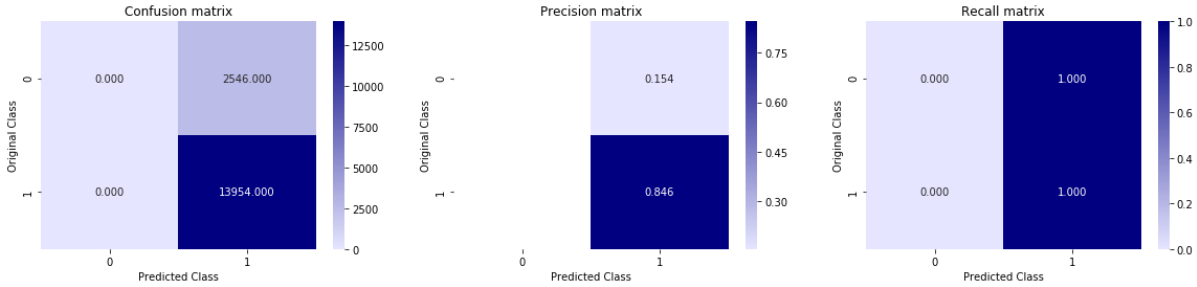
Observation: train auc=66.80%, test auc=60.98%

In [90]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```




```
Test confusion_matrix
```



Selecting 2000 best features.

In [0]:
```
from sklearn.preprocessing import Normalizer
```

train_nor=Normalizer(train_2) cv_nor=Normalizer(cv_2) test_nor=Normalizer(test_2)

In [82]:
```
#https://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection
#https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e
from sklearn.feature_selection import SelectKBest, chi2, f_classif

train_new = SelectKBest(score_func=f_classif, k=2000).fit_transform(train_2, project_data_y_train)
cv_new = SelectKBest(score_func=f_classif, k=2000).fit_transform(cv_2, project_data_y_cv)
test_new = SelectKBest(score_func=f_classif, k=2000).fit_transform(test_2, project_data_y_test)
train_new.shape
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/feature_selection/univariate_selection.py:114: UserWarning:

Features [0 0 0 0 0 0 0 0 0 0 0 0 0 0] are constant.

/usr/local/lib/python3.6/dist-packages/sklearn/feature_selection/univariate_selection.py:114: UserWarning:

Features [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] are constant.

/usr/local/lib/python3.6/dist-packages/sklearn/feature_selection/univariate_selection.py:114: UserWarning:

Features [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] are constant.
```

Out[82]:
```
(22445, 2000)
```

train_test_split

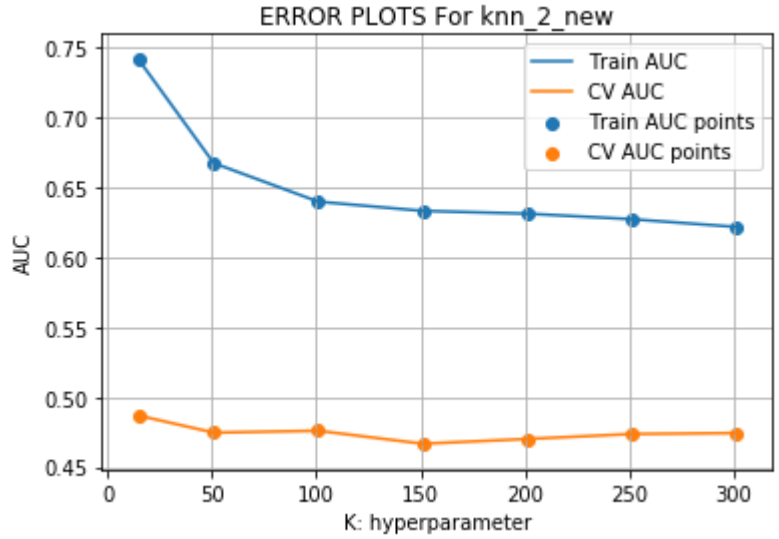
from sklearn.model_selection import train_test_split knn_2_new_train, knn_2_new_test, knn_2_new_y_train, knn_2_new_y_test = train_test_split(knn_2_new, y, test_size=0.33, stratify=y) knn_2_new_train, knn_2_new_cv, knn_2_new_y_train, knn_2_new_y_cv = train_test_split(knn_2_new_train, knn_2_new_y_train, test_size=0.33, stratify=knn_2_new_y_train)

https://medium.com/@plog397/auc-roc-curve-scoring-function-for-multi-class-classification-9822871a6659 (https://medium.com/@plog397/auc-roc-curve-scoring-function-for-multi-class-classification-9822871a6659)

def multiclass_roc_auc_score(y_test, y_pred, average="macro"): lb = LabelBinarizer() lb.fit(y_test) y_test = lb.transform(y_test) y_pred = lb.transform(y_pred) return roc_auc_score(y_test, y_pred, average=average)

In [83]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import LabelBinarizer     #https://stackoverflow.com/questions/31947140/sklearn-labelbinarizer-returns-vector-when-there-are-2-classes
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

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

"""


train_auc_2_new = []
cv_auc_2_new = []
K = [15, 51, 101, 151, 201, 251, 301]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, algorithm='brute')
    neigh.fit(train_new, project_data_y_train)

    y_train_pred = batch_predict(neigh, train_new)
    y_cv_pred = batch_predict(neigh, cv_new)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
    # not the predicted outputs
    train_auc_2_new.append(roc_auc_score(project_data_y_train,y_train_pred))
    cv_auc_2_new.append(roc_auc_score(project_data_y_cv, y_cv_pred))

plt.plot(K, train_auc_2_new, label='Train AUC')
plt.plot(K, cv_auc_2_new, label='CV AUC')

plt.scatter(K, train_auc_2_new, label='Train AUC points')
plt.scatter(K, cv_auc_2_new, label='CV AUC points')

plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS For knn_2_new")
plt.grid()
plt.show()
```

```
100%|██████████| 7/7 [07:12<00:00, 61.99s/it]
```



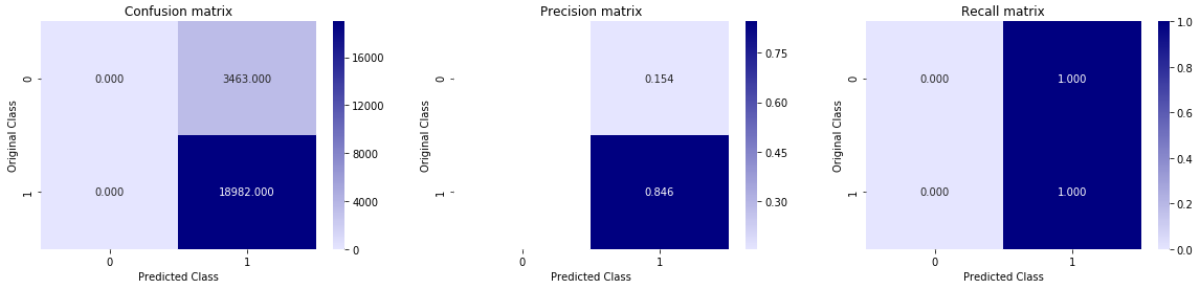
In [0]:
```
# from the error plot we choose K such that, we will have maximum AUC on cv da
ta and gap between the train and cv is less
# Note: based on the method you use you might get different hyperparameter val
ues as best one
# so, we choose according to the method you choose, you use gridsearch if you
 are having more computing power and note it will take more time
# if we increase the cv values in the GridSearchCV you will get more rebust re
sults.

#here we are choosing the best_k based on forloop results
best_k = 201
```

In [85]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


neigh = KNeighborsClassifier(n_neighbors=best_k, algorithm='brute')
neigh.fit(train_new, project_data_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, train_new)
y_test_pred = batch_predict(neigh, test_new)

train_fpr, train_tpr, tr_thresholds = roc_curve(project_data_y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(project_data_y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS for knn_2_new")
plt.grid()
plt.show()
```

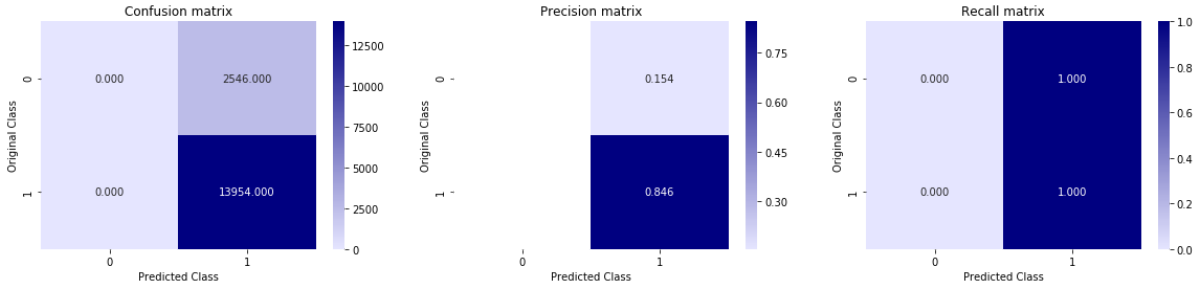
Observation: this shows a lot of error with only top 2000 features train auc=63.1%, test auc=44.7%

In [86]:
```
print('Train confusion_matrix')
plot_confusion_matrix(project_data_y_train,Zero1(y_train_pred))
print('Test confusion_matrix')
plot_confusion_matrix(project_data_y_test,Zero1(y_test_pred))
```

```
Train confusion_matrix
```




```
Test confusion_matrix
```



Observation: Here we can see taht test accuracy decreases when we select top 2000 features.

3. Conclusion

In [1]:
```
# Please compare all your models using Prettytable library
#https://stackoverflow.com/questions/18601688/python-prettytable-example
from prettytable import PrettyTable
x=PrettyTable()
x.field_names = ["Model", "Vectorizer", "train_auc", "test_auc", "best_k"]

x.add_row(["Knn", "Bow", "0.6635", "0.6260", "181"])
x.add_row(["Knn","tfidf", "0.6375", "0.6185", "201"])
x.add_row(["Knn","avgw2v", "0.6635", "0.6035", "81"])
x.add_row(["Knn","tfidfw2v", "0.6680", "0.60985", "71"])
x.add_row(["Knn","2000tfidf", "0.633", "0.4427", "201"])
print(x)
```

```
+-------+------------+-----------+----------+--------+
| Model | Vectorizer | train_auc | test_auc | best_k |
+-------+------------+-----------+----------+--------+
|  Knn  |    Bow     |   0.6635  |  0.6260  |  181   |
|  Knn  |   tfidf    |   0.6375  |  0.6185  |  201   |
|  Knn  |   avgw2v   |   0.6635  |  0.6035  |   81   |
|  Knn  |  tfidfw2v  |   0.6680  | 0.60985  |   71   |
|  Knn  | 2000tfidf  |   0.633   |  0.4427  |  201   |
+-------+------------+-----------+----------+--------+
```

In [0]:
```
#Steps followed:
#1. Importing all the necessary libraries
#2. merging project data and resource data into project data
#3. preprocessing of text features + sentiment scores
#4. dividing datasets into train test and split.
#5. categorical encoding
#6. numerical standardization
#7. bow+tfidf+avgw2v+tfidfw2v on text data
#8. stacking all vectorised features.
#9. Applying knn on different sets
#10. Selecting 2000 best features on set 2 and apply knn
#11. ploting auc curve and confusion matrix and preetytable
```