

DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	
<code>project_id</code>	A unique identifier for the proposed project.
<code>project_title</code>	Title of the
•	Art Will
<code>project_grade_category</code>	Grade level of students for which the project is targeted. C e
•	
•	
•	

Feature

	One or more (comma-separated) subject categories for the project.
project_subject_categories	<ul style="list-style-type: none"> • English Language Arts • Science • Social Studies • Math • Technology • Literacy & Language, Arts
school_state	State where school is located (Two-letter state abbreviations)
project_subject_subcategories	One or more (comma-separated) subject subcategories for the project.
project_resource_summary	<p>An explanation of the resources needed for the project.</p> <ul style="list-style-type: none"> • My students need hands on literacy materials.
project_essay_1	First essay.
project_essay_2	Second essay.
project_essay_3	Third essay.
project_essay_4	Fourth essay.
project_submitted_datetime	Datetime when project application was submitted. Example: 2018-07-08T12:00:00Z
teacher_id	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bf6
teacher_prefix	Teacher's title. One of the following entries: • Mr. • Ms. • Mrs. • Dr. • Prof.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the teacher.

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

```
In [1]: %matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

1.1 Reading Data

```
In [2]: project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

```
In [3]: print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)

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

```
In [4]: print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

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

1.2 preprocessing of project_subject_categories

```
In [5]: categories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-p
cat_list = []
for i in categories:
    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", "War
        if 'The' in j.split(): # this will split each of the category based on space
            j=j.replace('The','') # if we have the words "The" we are going to replace
            j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
            temp+=j.strip()+" "# abc ".strip() will return "abc", remove the trailing
            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 [6]: sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-list-of-strings-in-python/
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-list-of-strings-in-python/
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

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

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4033305
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.3 Text preprocessing

```
In [7]: # 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 [8]: project_data.head(2)
```

Out[8]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221 p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945 p258326	897464ce9ddc600bcfd1151f324dd63a	Mr.	FL	20

```
In [9]: ##### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

```
In [10]: # 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)
```

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 countries represented with the families within our school. Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, and respect.\r\nThe limits of your language are the limits of your world.\r\n-Ludwig Wittgenstein Our English learner's have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English along side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one at home is able to assist. All families with students within the Level 1 proficiency status, will be offered to be a part of this program. These educational videos will be specially chosen by the English Learner Teacher and will be sent home regularly to watch. The videos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\n\r\n====

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity. My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the students who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing

ing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still.nannan

How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an "open classroom" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more. With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

```
In [11]: # 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 [12]: sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("=*50)
```

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. They want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

```
In [13]: # \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\n', ' ')
sent = sent.replace('\\t', ' ')
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. The materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. They want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and fine motor skills. They also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

```
In [14]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays cognitive delays gross fine motor delays to autism They are eager beavers and always strive to work their hardest working past their limitations The materials we have are the ones I seek out for my students I teach in a Title I school where most of the students receive free or reduced price lunch Despite their disabilities and limitations my students love coming to school and come eager to learn and explore Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting This is how my kids feel all the time They want to be able to move as they learn or so they say Wobble chairs are the answer and I love them because they develop their core which enhances gross motor and in Turn fine motor skills They also want to learn through games my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Physical engagement is the key to our success The number toss and color and shape mats can make that happen My students will forget they are doing work and just have the fun a 6 year old deserves nannan

```
In [15]: # 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'the', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "di", "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'won', "won't", 'wouldn', "wouldn't"]
```

```
In [16]: # 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% |
| 109248/109248 [02:20<00:00, 778.77it/s]

1.4 Preprocessing of project_title

```
In [17]: # similarly you can preprocess the titles also
print(project_data['project_title'].values[0])
print("=*50")
print(project_data['project_title'].values[150])
print("=*50")
print(project_data['project_title'].values[1000])
print("=*50")
print(project_data['project_title'].values[10])
print("=*50")
print(project_data['project_title'].values[100])
print("=*50")
print(project_data['project_title'].values[1500])
print("=*50")
```

Educational Support for English Learners at Home
=====

More Movement with Hokki Stools
=====

Sailing Into a Super 4th Grade Year
=====

Reading Changes Lives
=====

21st Century learners, 21st century technology!
=====

Listening Center
=====

```
In [18]: preprocessed_titles = []

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

100% |██████████|
109248/109248 [00:06<00:00, 17364.65it/s]

```
In [19]: preprocessed_titles = pd.DataFrame(preprocessed_titles)
project_data['project_title'] = preprocessed_titles
project_data['project_title'].head(10)
```

```
Out[19]: 0      educational support english learners home
          1      wanted projector hungry learners
          2      soccer equipment awesome middle school students
          3      techie kindergarteners
          4      interactive math tools
          5      flexible seating mrs jarvis terrific third gra...
          6      chromebooks special education reading program
          7      it 21st century
          8      targeting more success class
          9      just for love reading pure pleasure
Name: project_title, dtype: object
```

```
In [20]: # https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).r
price_data.head(2)
```

Out[20]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

```
In [21]: # we cannot remove rows where teacher prefix is not available therefore we are re
# 'null'(string)
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-o
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('')
```

```
In [22]: # join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

```
In [23]: X_p = project_data[project_data['project_is_approved']==1].sample(frac = 0.2)
X_n = project_data[project_data['project_is_approved']==0].sample(frac = 1)
project_data = pd.concat([X_p,X_n])
```

```
In [24]: y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
X = project_data
```

```
In [25]: project_data.shape
```

```
Out[25]: (35083, 19)
```

Assignment 4: Naive Bayes

1. Apply Multinomial NaiveBayes on these feature sets

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

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

2. The hyper parameter tuning(find best Alpha)

- Find the best hyper parameter which will give the maximum AUC (<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/>) value
- Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
- Find the best hyper parameter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. Feature importance

- Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names

4. Representation of results

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.



- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.



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



(<https://seaborn.pydata.org/generated/seaborn.heatmap.html>)

5. Conclusion

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



2. Naive Bayes

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

```
In [26]: #https://pythonprogrammingLanguage.com/training-and-test-data/
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
X_train , X_cv, y_train, y_cv = train_test_split(X_train,y_train,test_size=0.3)
```

```
In [27]: print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
```

```
(17190, 19) (17190,)
(7368, 19) (7368,)
(10525, 19) (10525,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

```
In [28]: # please write all the code with proper documentation, and proper titles for each
# 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
# make sure you featurize train and test data separately

# 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

# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer1 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=True)
vectorizer1.fit(X_train['clean_categories'].values)
print(vectorizer1.get_feature_names())

categories_one_hot_train = vectorizer1.transform(X_train['clean_categories'].values)
print("Shape of matrix after one hot encoding ", categories_one_hot_train.shape)
categories_one_hot_test = vectorizer1.transform(X_test['clean_categories'].values)
print("Shape of matrix after one hot encoding ", categories_one_hot_test.shape)
categories_one_hot_cv = vectorizer1.transform(X_cv['clean_categories'].values)
print("Shape of matrix after one hot encoding ", categories_one_hot_cv.shape)

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding (17190, 9)
Shape of matrix after one hot encoding (10525, 9)
Shape of matrix after one hot encoding (7368, 9)
```

```
In [29]: vectorizer1.get_feature_names()
```

```
Out[29]: ['Warmth',
          'Care_Hunger',
          'History_Civics',
          'Music_Arts',
          'AppliedLearning',
          'SpecialNeeds',
          'Health_Sports',
          'Math_Science',
          'Literacy_Language']
```

```
In [30]: # we use count vectorizer to convert the values into one hot encoded features
vectorizer2 = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=True)
vectorizer2.fit(X_train['clean_subcategories'].values)
print(vectorizer2.get_feature_names())
```

```
sub_categories_one_hot_train = vectorizer2.transform(X_train['clean_subcategories'])
print("Shape of matrix after one hot encoding ", sub_categories_one_hot_train.shape)
sub_categories_one_hot_cv = vectorizer2.transform(X_cv['clean_subcategories'].values)
print("Shape of matrix after one hot encoding ", sub_categories_one_hot_cv.shape)
sub_categories_one_hot_test = vectorizer2.transform(X_test['clean_subcategories'].values)
print("Shape of matrix after one hot encoding ", sub_categories_one_hot_test.shape)

['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encoding (17190, 30)
Shape of matrix after one hot encoding (7368, 30)
Shape of matrix after one hot encoding (10525, 30)
```

```
In [31]: vectorizer2.get_feature_names()
```

```
Out[31]: ['Economics',
 'CommunityService',
 'FinancialLiteracy',
 'ParentInvolvement',
 'Extracurricular',
 'Civics_Government',
 'ForeignLanguages',
 'NutritionEducation',
 'Warmth',
 'Care_Hunger',
 'SocialSciences',
 'PerformingArts',
 'CharacterEducation',
 'TeamSports',
 'Other',
 'College_CareerPrep',
 'Music',
 'History_Geography',
 'Health_LifeScience',
 'EarlyDevelopment',
 'ESL',
 'Gym_Fitness',
 'EnvironmentalScience',
 'VisualArts',
 'Health_Wellness',
 'AppliedSciences',
 'SpecialNeeds',
 'Literature_Writing',
 'Mathematics',
 'Literacy']
```

```
In [32]: # Please do the similar feature encoding with state, teacher_prefix and project_g
# https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/
vectorizer3 = CountVectorizer()
vectorizer3.fit(X_train['school_state'].values) # fit has to happen only on train

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer3.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer3.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer3.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer3.get_feature_names())
```

After vectorizations
(17190, 51) (17190,)
(7368, 51) (7368,)
(10525, 51) (10525,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia',
'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms',
'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', 'pa',
'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']

```
In [33]: project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('null')
```

```
In [34]: # Please do the similar feature encoding with state, teacher_prefix and project_g
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-value-error
vectorizer4 = CountVectorizer()
vectorizer4.fit(X_train['teacher_prefix'].values.astype('U')) # fit has to happen only on train

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_onehotencode = vectorizer4.transform(X_train['teacher_prefix'].values.astype('U'))
X_cv_teacher_onehotencode = vectorizer4.transform(X_cv['teacher_prefix'].values.astype('U'))
X_test_teacher_onehotencode = vectorizer4.transform(X_test['teacher_prefix'].values.astype('U'))

print("After vectorizations")
print(X_train_teacher_onehotencode.shape, y_train.shape)
print(X_cv_teacher_onehotencode.shape, y_cv.shape)
print(X_test_teacher_onehotencode.shape, y_test.shape)
print(vectorizer4.get_feature_names())
```

After vectorizations
(17190, 5) (17190,)
(7368, 5) (7368,)
(10525, 5) (10525,)
['dr', 'mr', 'mrs', 'ms', 'teacher']

```
In [35]: #This step is to intialize a vectorizer with vocab from train data
from collections import Counter
my_counter4 = Counter()
for word in X_train['project_grade_category'].values:
    my_counter4.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grade_category_dict = dict(my_counter4)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items))
```

```
In [36]:
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

values = (np.array(X_test['project_grade_category']))
print(values[1:10])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:10])

# binary encode

project_grade_category_onehot_encoder_test = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
project_grade_category_onehot_encoder_test = project_grade_category_onehot_encoder_test.fit_transform(integer_encoded)

print("Shape of matrix after one hot encoding ",project_grade_category_onehot_encoder_test.shape)
print(['Grades 3-5' 'Grades PreK-2' 'Grades 3-5' 'Grades PreK-2' 'Grades 6-8'
      'Grades PreK-2' 'Grades 3-5' 'Grades PreK-2' 'Grades PreK-2'])
print([0 3 0 3 1 3 0 3 3])
Shape of matrix after one hot encoding  (10525, 4)
```

In [37]: # binary encode

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

values = (np.array(X_train['project_grade_category']))
print(values[1:10])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:10])

# binary encode

project_grade_category_onehot_encoder_train = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
project_grade_category_onehot_encoder_train = project_grade_category_onehot_encoder_train.fit(integer_encoded)

print("Shape of matrix after one hot encoding ", project_grade_category_onehot_encoder_train.transform(values).shape)
['Grades PreK-2' 'Grades PreK-2' 'Grades PreK-2' 'Grades 3-5'
 'Grades PreK-2' 'Grades PreK-2' 'Grades 3-5' 'Grades 6-8' 'Grades 3-5']
[3 3 3 0 3 3 0 1 0]
Shape of matrix after one hot encoding (17190, 4)
```

In [38]: # binary encode

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

values = (np.array(X_cv['project_grade_category']))
print(values[1:10])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:10])

# binary encode

project_grade_category_onehot_encoder_cv = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
project_grade_category_onehot_encoder_cv = project_grade_category_onehot_encoder_cv.fit(integer_encoded)

print("Shape of matrix after one hot encoding ", project_grade_category_onehot_encoder_cv.transform(values).shape)
['Grades 3-5' 'Grades 6-8' 'Grades 9-12' 'Grades 3-5' 'Grades 9-12'
 'Grades PreK-2' 'Grades 9-12' 'Grades 3-5' 'Grades PreK-2']
[0 1 2 0 2 3 2 0 3]
Shape of matrix after one hot encoding (7368, 4)
```

In [39]: vectorizer5 = ['Grades PreK-2', 'Grades 3-5', 'Grades 9-12', 'Grades 6-8']

```
In [69]: #https://scikit-Learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html
from sklearn.preprocessing import Normalizer
price = X_train['price']
price = pd.DataFrame(price)
price_normalized_train = Normalizer().fit_transform(price)

price = X_test['price']
price = pd.DataFrame(price)
price_normalized_test = Normalizer().fit_transform(price)

from sklearn.preprocessing import Normalizer
price = X_cv['price']
price = pd.DataFrame(price)
price_normalized_cv = Normalizer().fit_transform(price)
```

```
In [70]: print(price_normalized_train.shape)
print(price_normalized_test.shape)
print(price_normalized_cv.shape)
```

```
(17190, 1)
(10525, 1)
(7368, 1)
```

```
In [71]: #https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-validation
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['essay'].values)
text_tfidf_train = vectorizer.transform(X_train['essay'])
print("Shape of matrix after one hot encoding ",text_tfidf_train.shape)
text_tfidf_test = vectorizer.transform(X_test['essay'])
print("Shape of matrix after one hot encoding ",text_tfidf_test.shape)
text_tfidf_cv = vectorizer.transform(X_cv['essay'])
print("Shape of matrix after one hot encoding ",text_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (17190, 8108)
Shape of matrix after one hot encoding  (10525, 8108)
Shape of matrix after one hot encoding  (7368, 8108)
```

```
In [72]: vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['project_title'].values.astype('U'))
title_tfidf_train = vectorizer.transform(X_train['project_title'].values.astype('U'))
print("Shape of matrix after one hot encoding ",title_tfidf_train.shape)
title_tfidf_test = vectorizer.transform(X_test['project_title'].values.astype('U'))
print("Shape of matrix after one hot encoding ",title_tfidf_test.shape)
title_tfidf_cv = vectorizer.transform(X_cv['project_title'].values.astype('U'))
print("Shape of matrix after one hot encoding ",title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (17190, 1016)
Shape of matrix after one hot encoding  (10525, 1016)
Shape of matrix after one hot encoding  (7368, 1016)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [73]:

```

# please write all the code with proper documentation, and proper titles for each
# 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
# make sure you featurize train and test data separately

# 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
from sklearn.feature_extraction.text import CountVectorizer
vectorizer6 = CountVectorizer(min_df=10)
vectorizer6.fit(X_train['essay'].values)

text_bow_train = vectorizer6.transform(X_train['essay'].values)
text_bow_cv = vectorizer6.transform(X_cv['essay'].values)
text_bow_test = vectorizer6.transform(X_test['essay'].values)

print(text_bow_train.shape, y_train.shape)
print(text_bow_cv.shape, y_cv.shape)
print(text_bow_test.shape, y_test.shape)

```

(17190, 8108) (17190,)
(7368, 8108) (7368,)
(10525, 8108) (10525,)

In [74]:

```

vectorizer7 = CountVectorizer(min_df=10)
vectorizer7.fit(X_train['project_title'].values.astype('U'))

title_bow_train = vectorizer7.transform(X_train['project_title'].values.astype('U'))
title_bow_cv = vectorizer7.transform(X_cv['project_title'].values.astype('U'))
title_bow_test = vectorizer7.transform(X_test['project_title'].values.astype('U'))

print(title_bow_train.shape, y_train.shape)
print(title_bow_cv.shape, y_cv.shape)
print(title_bow_test.shape, y_test.shape)

```

(17190, 1016) (17190,)
(7368, 1016) (7368,)
(10525, 1016) (10525,)

In [75]:

```

...
# 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 preproc_texts:
    words.extend(i.split(' '))

for i in preproc_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

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

...

```

Out[75]: '\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084

```
039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGloveModel(glo  
veFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\\'r\\', enco  
ding="utf8")\n    model = {} \n    for line in tqdm(f):\n        splitLine = lin  
e.split()\n        word = splitLine[0]\n        embedding = np.array([float(va  
l) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model), " words loaded!")\n    return model\nmodel = loadGloveModel('\\gl  
ove.42B.300d.txt')\n\n# =====\nOutput:\n\nLoading G  
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495 words loaded!\n\n# =  
=====\\n\\nwords = []\nfor i in preproc_texts:\n    word  
s.extend(i.split(' '))\nfor i in preproc_titles:\n    words.extend(i.spli  
t(' '))\nprint("all the words in the coupus", len(words))\nwords = set(words)  
\\nprint("the unique words in the coupus", len(words))\n\\ninter_words = set(mode  
l.keys()).intersection(words)\nprint("The number of words that are present in b  
oth glove vectors and our coupus", len(inter_words),",",np.round(len(inte  
r_words)/len(words)*100,3), "%")\n\\nwords_courpus = {}\nwords_glove = set(mode  
l.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n\\n\\n# stronging va  
riables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-  
to-save-and-load-variables-in-python/\n\\nimport (http://www.jessicayung.com/how-  
-to-use-pickle-to-save-and-load-variables-in-python/\n\\nimport) pickle\\nwith op  
en('\\glove_vectors\\', '\\wb\\') as f:\\n    pickle.dump(words_courpus, f)\n\\n\\n'
```

```
In [76]: # stronging variables into pickle files python: http://www.jessicayung.com/how-to  
# make sure you have the glove_vectors file  
with open('glove_vectors', 'rb') as f:  
    model = pickle.load(f)  
    glove_words = set(model.keys())
```

```
In [77]: # average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train = [] # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero Length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

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


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

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


avg_w2v_vectors_cv = [] # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero Length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

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

100%|██████████| 17190/17190 [00:14<00:00, 1164.68it/s]

17190

300

100%|██████████| 10525/10525 [00:10<00:00, 995.19it/s]

10525

300

100% |

███████████ 7368/7368 [00:06<00:00, 1060.85it/s]

7368

300

```
In [78]: # average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train_title = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_train['project_title'].values.astype('U')): # for each review
    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_title.append(vector)

print(len(avg_w2v_vectors_train_title))
print(len(avg_w2v_vectors_train_title[0]))


# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_test_title = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_test['project_title'].values.astype('U')): # for each review
    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_title.append(vector)

print(len(avg_w2v_vectors_test_title))
print(len(avg_w2v_vectors_test_title[0]))


# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_cv_title = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_cv['project_title'].values.astype('U')): # for each review
    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_title.append(vector)

print(len(avg_w2v_vectors_cv_title))
print(len(avg_w2v_vectors_cv_title[0]))
```

100% | 
| 17190/17190 [00:00<00:00, 30333.98it/s]

17190
300

```
100%|██████████| 10525/10525 [00:00<00:00, 27640.65it/s]
```

10525
300

```
100%|██████████| 7368/7368 [00:00<00:00, 31505.38it/s]
```

7368
300

2.4 Applying NB() on different kind of featurization as mentioned in the instructions

Apply Naive Bayes 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

2.4.1 Applying Naive Bayes on BOW, SET 1

In [79]:

```
teacher_onehotencode, project_grade_category_onehot_encoder_train, price_normalized_
encoder_onehotencode, project_grade_category_onehot_encoder_test, price_normalized_test_
tencode, project_grade_category_onehot_encoder_cv, price_normalized_cv, text_bow_c
```

Final Data matrix
(17190, 9224) (17190,)
(7368, 9224) (7368,)
(10525, 9224) (10525,)

In [80]:

```
print(X1.shape)
print(y_train.shape)
```



```
(17190, 9224)
(17190,)
```

```
In [81]: def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability esti
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1
    # in this for Loop we will iterate until 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 [82]: %%time
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
alpha = [10**x for x in range(-4,4)]
for i in alpha:
    nb = MultinomialNB(alpha=i, fit_prior=False)
    nb.fit(X1, y_train)

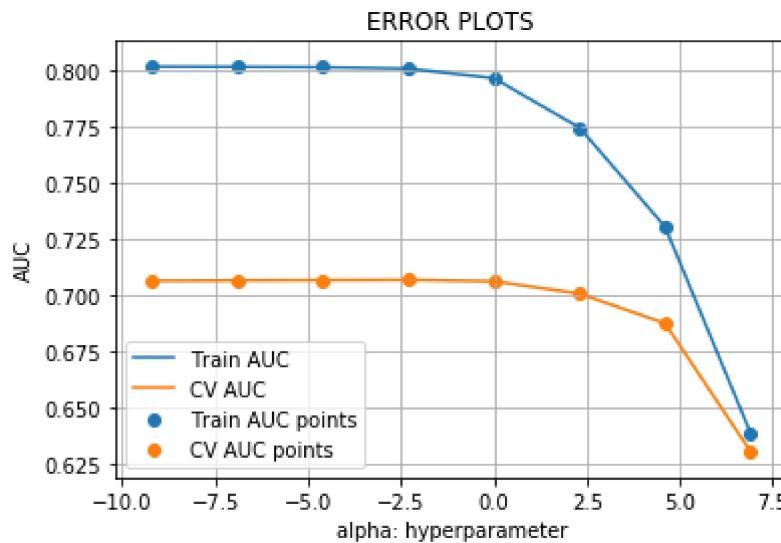
    y_train_pred = batch_predict(nb, X1)
    y_cv_pred = batch_predict(nb, X3)

    train_auc.append(roc_auc_score(y_train, y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(np.log(alpha), train_auc, label='Train AUC')
plt.plot(np.log(alpha), cv_auc, label='CV AUC')

plt.scatter(np.log(alpha), train_auc, label='Train AUC points')
plt.scatter(np.log(alpha), cv_auc, label='CV AUC points')

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



Wall time: 1.98 s

```
In [83]: %%time
# https://scikit-Learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
import math
nb = MultinomialNB(alpha=i, fit_prior=False)
parameters = {'alpha':[(10**x) for x in range(-4,4)]}
clf = GridSearchCV(nb, parameters, cv=3, scoring='roc_auc', return_train_score=True)
clf.fit(X1, y_train)

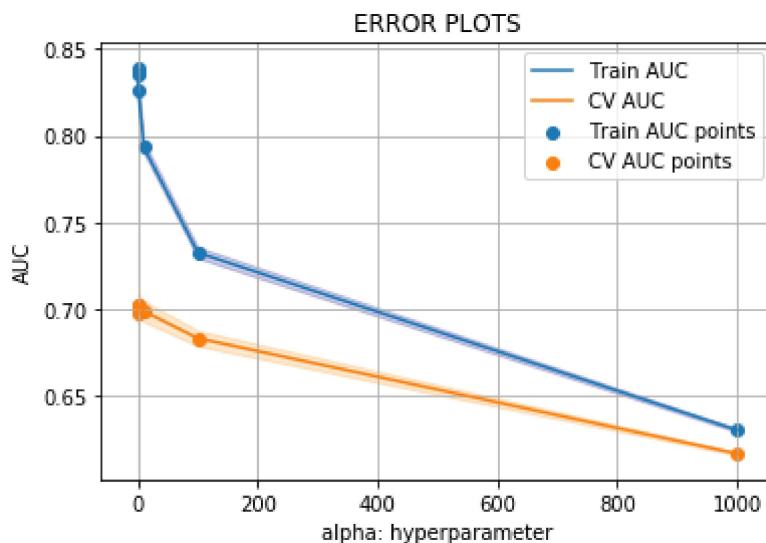
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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

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

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

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



Wall time: 3.32 s

```
In [84]: type(parameters)
```

```
Out[84]: dict
```

```
In [85]: best_alpha = 40
```

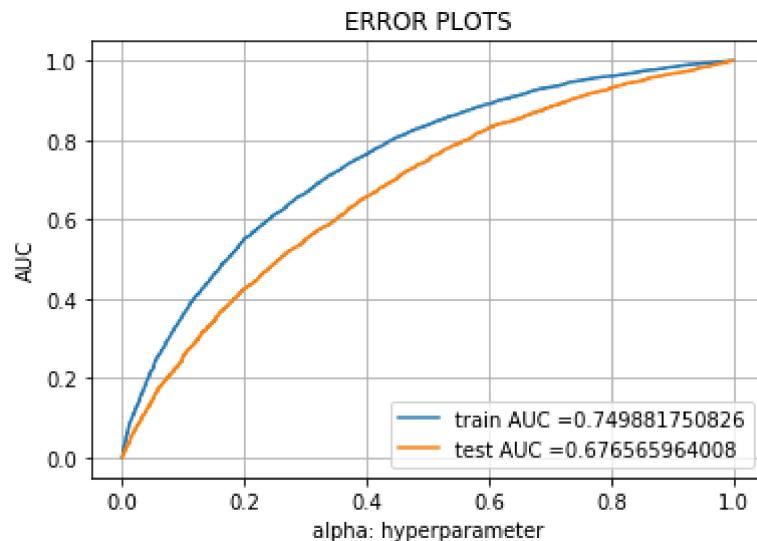
```
In [86]: from sklearn.metrics import roc_curve, auc
```

```
nb = MultinomialNB(alpha = best_alpha, fit_prior=False)
nb.fit(X1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimate.
# not the predicted outputs

y_train_pred = batch_predict(nb, X1)
y_test_pred = batch_predict(nb, X2)

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

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



```
In [87]: # we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshold, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold",
predictions = []
for i in proba:
    if i>=t:
        predictions.append(1)
    else:
        predictions.append(0)
return predictions
```

```
In [88]: print("*100)
from sklearn.metrics import confusion_matrix
#https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

y_pred_new = nb.predict(X1)

print("Confusion Matrix of train set:\n [ [TN  FP]\n [FN  TP] ]\n")
df_cm = pd.DataFrame(confusion_matrix(y_train, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')

=====
=====

Confusion Matrix of train set:
[ [TN  FP]
[FN  TP] ]
```

Out[88]: <matplotlib.axes._subplots.AxesSubplot at 0x20b41228da0>



```
In [89]: y_pred_new = nb.predict(X2)

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

Out[89]: <matplotlib.axes._subplots.AxesSubplot at 0x20b47df2048>



2.4.1.1 Top 10 important features of positive class from SET 1

```
In [90]: all_features=[]
```

```
In [91]: all_features.extend(vectorizer1.get_feature_names())
all_features.extend(vectorizer2.get_feature_names())
all_features.extend(vectorizer3.get_feature_names())
all_features.extend(vectorizer4.get_feature_names())
all_features.extend(vectorizer5)
all_features.extend(vectorizer6.get_feature_names())
all_features.extend(vectorizer7.get_feature_names())
all_features.append('price')
all_features.append('quantity')
all_features.append('teacher_number_of_previously_posted_projects')
```

```
In [92]: #https://stats.stackexchange.com/questions/266031/what-is-log-probability-of-feature
pos_class_prob_sorted = np.argsort((nb.feature_log_prob_)[1])
print(pos_class_prob_sorted[:10])
```

[5565 7966 5346 7661 3695 4992 90 7685 3412 8344]

```
In [93]: all_features = np.array(all_features)
```

```
In [94]: np.take(all_features,pos_class_prob_sorted[:10])
```

```
Out[94]: array(['philadelphia', 'wars', 'outs', 'trimester', 'hydrated', 'nhere',
       'dr', 'tummies', 'group', 'can'],
      dtype='<U44')
```

2.4.1.2 Top 10 important features of negative class from SET 1

```
In [95]: neg_class_prob_sorted = np.argsort((nb.feature_log_prob_)[0])
```

```
In [96]: # Please write all the code with proper documentation
np.take(all_features,neg_class_prob_sorted[:10])
```

```
Out[96]: array(['nhow', 'centers', 'gain', 'warm', 'convey', 'matches', 'bunch',
       'thrill', 'clarity', 'pink'],
      dtype='<U44')
```

2.4.2 Applying Naive Bayes on TFIDF, SET 2

```
In [97]: from scipy.sparse import hstack
X4 = hstack((categories_one_hot_train, sub_categories_one_hot_train, X_train_stat)
X5 = hstack((categories_one_hot_test, sub_categories_one_hot_test, X_test_state_o
X6 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, X_cv_state_ohe, X_
print("Final Data matrix")
print(X1.shape, y_train.shape)
print(X3.shape, y_cv.shape)
print(X2.shape, y_test.shape)
```

Final Data matrix
(17190, 9224) (17190,)
(7368, 9224) (7368,)
(10525, 9224) (10525,)

```
In [98]: def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability esti
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000
    # in this for Loop we will iterate until 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 [99]: %%time
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
alpha = [10**x for x in range(-4,4)]
for i in alpha:
    nb = MultinomialNB(alpha=i, fit_prior=False)
    nb.fit(X4, y_train)

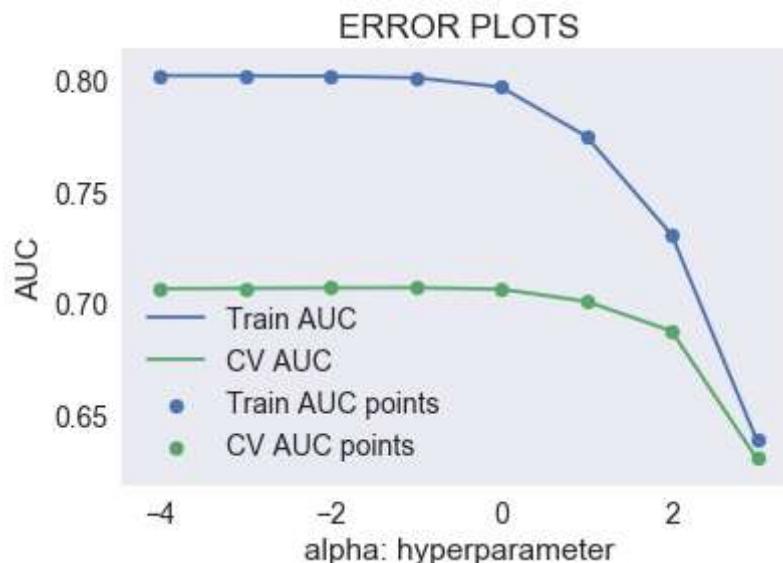
    y_train_pred = batch_predict(nb, X4)
    y_cv_pred = batch_predict(nb, X6)

    train_auc.append(roc_auc_score(y_train, y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

plt.plot(np.log10(alpha), train_auc, label='Train AUC')
plt.plot(np.log10(alpha), cv_auc, label='CV AUC')

plt.scatter(np.log10(alpha), train_auc, label='Train AUC points')
plt.scatter(np.log10(alpha), cv_auc, label='CV AUC points')

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



Wall time: 2.27 s

```
In [100]: %%time
# https://scikit-Learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB(alpha=i, fit_prior=False)
parameters = {'alpha':[10**x for x in range(-4,4)]}
clf = GridSearchCV(nb, parameters, cv=3, scoring='roc_auc', return_train_score=True)
clf.fit(X4, y_train)

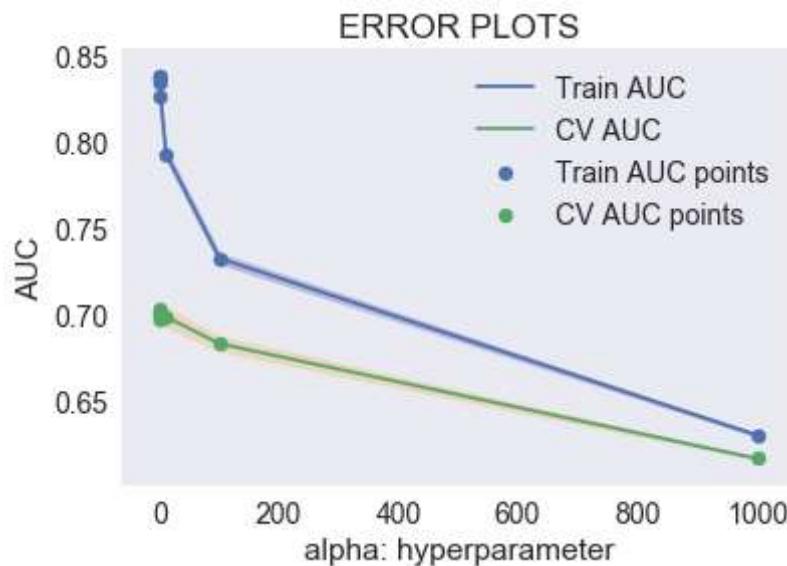
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

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

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

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

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



Wall time: 3.7 s

```
In [101]: best_alpha = 40
```

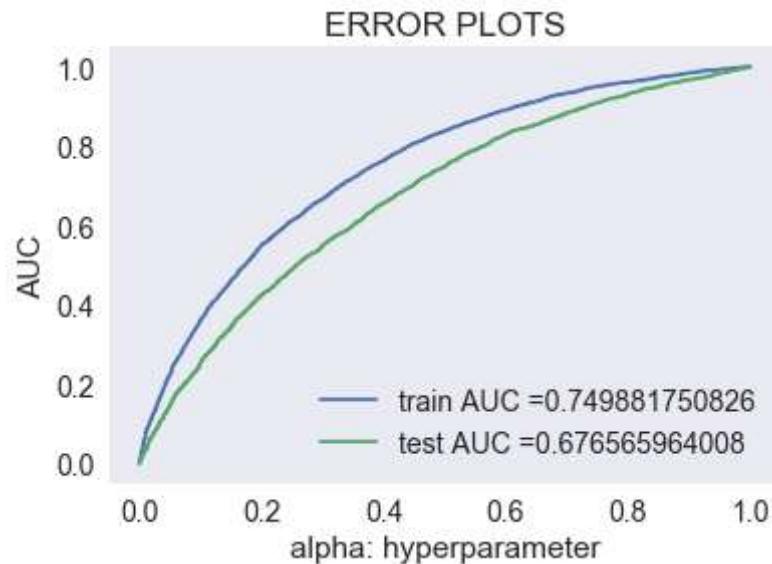
```
In [102]: from sklearn.metrics import roc_curve, auc
```

```
nb = MultinomialNB(alpha = best_alpha, fit_prior=False)
nb.fit(X4, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimate.
# not the predicted outputs

y_train_pred = batch_predict(nb, X4)
y_test_pred = batch_predict(nb, X5)

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

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



```
In [103]: # we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshold, fpr, tpr):

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold",
predictions = []
for i in proba:
    if i>=t:
        predictions.append(1)
    else:
        predictions.append(0)
return predictions
```

```
In [104]: print("*100)
from sklearn.metrics import confusion_matrix
#https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

y_pred_new = nb.predict(X4)

print("Confusion Matrix of train set:\n [ [TN  FP]\n [FN  TP] ]\n")
df_cm = pd.DataFrame(confusion_matrix(y_train, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')

=====
=====

Confusion Matrix of train set:
[ [TN  FP]
[FN  TP] ]
```

Out[104]: <matplotlib.axes._subplots.AxesSubplot at 0x20b58ee1a90>



```
In [105]: y_pred_new = nb.predict(X5)
```

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

```
Out[105]: <matplotlib.axes._subplots.AxesSubplot at 0x20b471f9978>
```



3. Conclusions

```
In [106]: # Please compare all your models using Prettytable Library
# Please compare all your models using Prettytable Library
from prettytable import PrettyTable
```

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

```
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter('alpha')'", "AUC"]
x.add_row(['BOW', 'Naive Bayes', 40, 0.7421])
x.add_row(['TF-IDF', 'Naive Bayes', 40, 0.7473])
print(x)
```

Vectorizer	Model	Hyper Parameter('alpha')'	AUC
BOW	Naive Bayes	40	0.7421
TF-IDF	Naive Bayes	40	0.7473