

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>	<ul style="list-style-type: none">••	Title of the project Art Will
<code>project_grade_category</code>	<ul style="list-style-type: none">••••	Grade level of students for which the project is targeted

Feature	
	One or more (comma-separated) subject categories from the following enumeration:
	<ul style="list-style-type: none"> • • • • • • • •
project_subject_categories	Li
	<ul style="list-style-type: none"> • •
school_state	State where school is located (Two-letter U.S. state abbreviations)
	One or more (comma-separated) subject subcategories
project_subject_subcategories	<ul style="list-style-type: none"> • •
	Literature & Writing
	An explanation of the resources needed for the project.
project_resource_summary	<ul style="list-style-type: none"> • My students need hands on literacy materials.
project_essay_1	F
project_essay_2	Sec
project_essay_3	TI
project_essay_4	For
project_submitted_datetime	Datetime when project application was submitted. Example: 2015-01-01 12:00:00
teacher_id	A unique identifier for the teacher of the project. Example: bdf8baa8fedef6b
	Teacher's title. One of the following:
	<ul style="list-style-type: none"> • • • • • •
teacher_prefix	
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 <code>project_id</code> value from the <code>train.csv</code> 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
<code>project_is_approved</code>	A binary flag indicating whether DonorsChoose approved the project. A value of <code>0</code> indicates the project was not approved, and a value of <code>1</code> 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
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

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

from tqdm import tqdm
import os

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

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

Reading Data

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

```
In [3]: project_data.isnull().sum()
```

```
Out[3]: Unnamed: 0      0
id      0
teacher_id  0
teacher_prefix  3
school_state  0
project_submitted_datetime  0
project_grade_category  0
project_subject_categories  0
project_subject_subcategories  0
project_title  0
project_essay_1  0
project_essay_2  0
project_essay_3  105490
project_essay_4  105490
project_resource_summary  0
teacher_number_of_previously_posted_projects  0
project_is_approved  0
dtype: int64
```

```
In [4]: project_data.dropna(subset = ['teacher_prefix'], inplace=True)
```

```
In [5]: project_data.isnull().sum()
```

```
Out[5]: Unnamed: 0      0
id      0
teacher_id  0
teacher_prefix  0
school_state  0
project_submitted_datetime  0
project_grade_category  0
project_subject_categories  0
project_subject_subcategories  0
project_title  0
project_essay_1  0
project_essay_2  0
project_essay_3  105488
project_essay_4  105488
project_resource_summary  0
teacher_number_of_previously_posted_projects  0
project_is_approved  0
dtype: int64
```

```
In [6]: y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
```

In [7]: `project_data.head()`

Out[7]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	20
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	20
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	20

In [8]: `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 (109245, 16)

 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']

In [9]: `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[9]:

	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

Preprocessing of project_subject_categories

```
In [10]: 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
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-
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",
        if 'The' in j.split(): # this will split each of the category based on sp
            j=j.replace('The', '') # if we have the words "The" we are going to re
        j = j.replace(' ', '') # we are placing all the ' '(space) with ''(empty)
        temp+=j.strip()+" " # " abc ".strip() will return "abc", remove the trail
        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]))
```

Preprocessing of project_subject_subcategories

```

In [11]: sub_categories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/22898595/4000000

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

sub_cat_list = []
for i in sub_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", "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 string)
        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/22898595/4000000
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]))

```

Text preprocessing

```

In [12]: # 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 [13]: # Dropping the other 4 columns related in project essay
project_data.drop(['project_essay_1', 'project_essay_2', 'project_essay_3', 'project_essay_4'], axis=1, inplace=True)

```



```
In [14]: project_data.head(2)
```

Out[14]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

```
In [15]: # 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)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[29500])
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. \"The limits of your language are the limits of your world.\" -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\n By 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\n Parents 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\nnnnnnn

=====

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\n Whenever 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 on

e 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 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

=====

My wonderful students are 3, 4, and 5 years old. We are located in a small town outside of Charlotte, NC. All of my 22 students are children of school district employees.\r\nMy students are bright, energetic, and they love to learn! They love hands-on activities that get them moving. Like most preschoolers, they enjoy music and creating different things. \r\nAll of my students come from wonderful families that are very supportive of our classroom. Our parents enjoy watching their children's growth as much as we do! These materials will help me teach my students all about the life cycle of a butterfly. We will watch as the Painted Lady caterpillars grow bigger and build their chrysalis. After a few weeks they will emerge from the chrysalis as beautiful butterflies! We already have a net for the chrysalises, but we still need the caterpillars and feeding station.\r\nThis will be an unforgettable experience for my students. My student absolutely love hands-on materials. They learn so much from getting to touch and manipulate different things. The supporting materials I have selected will help my students understand the life cycle through exploration.nannan

=====

My students are an amazing group of eclectic children, coming from all walks of life. Many are from socioeconomically challenged homes, many from migrant families. The city is small so that most students who are permanent residents have known each other forever. It is a 'large family' of sorts. They all support each other and strive everyday to be successful. And they are! \r\nAs second language learners, many struggle day to day to learn in the classroom but

excel in physical activity! Most students think of exercise during the day as their recess time. By teaching them how to purposefully exercise, how to keep track of their exercise, as well as hypothesize results, students will create a lifelong love of exercise and health. My students told me how much they enjoy Physical Education outdoors. They have asked for field cones and activities such as fitness dice and foam rings to organize meaningful activities. These journals will be used to chart patterns and see growth. My students showed interest in my personal fitness tracker I wear. My students asked me to get them a set to track their fitness and give them data to chart for their math journals. \r\n\r\nPurposeful exercise not only creates a healthier body, but also instills a healthier mindset about exercise and lifelong health.nannan
=====

In [16]: [# https://stackoverflow.com/a/47091490/4084039](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)
    phrase = re.sub(r"%", " percent", phrase)
    return phrase
```

In [17]: `sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("=*50)`

Describing my students is not an easy task. Many would say that they are inspirational, creative, and hard-working. They are all unique - unique in their interests, their learning, their abilities, and so much more. What they all have in common is their desire to learn each day, despite difficulties that they encounter. \r\nOur classroom is amazing - because we understand that everyone learns at their own pace. As the teacher, I pride myself in making sure my students are always engaged, motivated, and inspired to create their own learning! \r\nThis project is to help my students choose seating that is more appropriate for them, developmentally. Many students tire of sitting in chairs during lessons, and having different seats available helps to keep them engaged and learning.\r\nFlexible seating is important in our classroom, as many of our students struggle with attention, focus, and engagement. We currently have stability balls for seating, as well as regular chairs, but these stools will help students who have trouble with balance, or find it difficult to sit on a stability ball for a long period of time. We are excited to try these stools as a part of our engaging classroom community!nannan
=====

```
In [18]: # \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)
```

Describing my students is not an easy task. Many would say that they are inspirational, creative, and hard-working. They are all unique - unique in their interests, their learning, their abilities, and so much more. What they all have in common is their desire to learn each day, despite difficulties that they encounter. Our classroom is amazing - because we understand that everyone learns at their own pace. As the teacher, I pride myself in making sure my students are always engaged, motivated, and inspired to create their own learning! This project is to help my students choose seating that is more appropriate for them, developmentally. Many students tire of sitting in chairs during lessons, and having different seats available helps to keep them engaged and learning. Flexible seating is important in our classroom, as many of our students struggle with attention, focus, and engagement. We currently have stability balls for seating, as well as regular chairs, but these stools will help students who have trouble with balance, or find it difficult to sit on a stability ball for a long period of time. We are excited to try these stools as a part of our engaging classroom community!

```
In [19]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

Describing my students is not an easy task Many would say that they are inspirational creative and hard working They are all unique unique in their interests their learning their abilities and so much more What they all have in common is their desire to learn each day despite difficulties that they encounter Our classroom is amazing because we understand that everyone learns at their own pace As the teacher I pride myself in making sure my students are always engaged motivated and inspired to create their own learning This project is to help my students choose seating that is more appropriate for them developmentally Many students tire of sitting in chairs during lessons and having different seats available helps to keep them engaged and learning Flexible seating is important in our classroom as many of our students struggle with attention focus and engagement We currently have stability balls for seating as well as regular chairs but these stools will help students who have trouble with balance or find it difficult to sit on a stability ball for a long period of time We are excited to try these stools as a part of our engaging classroom community

```
In [20]: project_data.shape
```

```
Out[20]: (109245, 13)
```

```
In [21]: # 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', 'itse',
            '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', 'th',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of',
            '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', "d",
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn",
            'won', "won't", 'wouldn', "wouldn't"]
```

```
In [22]: # Combining all the above snippets
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\r', ' ')
    sent = sent.replace('\n', ' ')
    sent = sent.replace('\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████|  
109245/109245 [01:31<00:00, 1193.63it/s]
```

```
In [23]: # after preprocessing
preprocessed_essays[20000]
```

```
Out[23]: 'wonderful students 3 4 5 years old located small town outside charlotte nc 22
students children school district employees students bright energetic love learn
love hands activities get moving like preschoolers enjoy music creating different
things students come wonderful families supportive classroom parents enjoy
watching children growth much materials help teach students life cycle butterfly
watch painted lady caterpillars grow bigger build chrysalis weeks emerge chrysalis
beautiful butterflies already not chrysalises still need caterpillars feeding
station unforgettable experience students student absolutely love hands materials
learn much getting touch manipulate different things supporting materials
selected help students understand life cycle exploration nannan'
```

Preprocessing of `project_title`

```
In [24]: # similarly preprocessing the titles also
preprocessed_title = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].values.astype('U')):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 1
09245/109245 [00:04<00:00, 25954.83it/s]
```

```
In [25]: project_data['project_title'] = pd.DataFrame(preprocessed_title)
```

```
In [26]: #Removing '.' from teacher prefix(as a process of text preprocessing)
project_data['teacher_prefix']=project_data['teacher_prefix'].str.replace('\.', '')
```

```
In [27]: project_data['teacher_prefix'].isna().any()
```

```
Out[27]: False
```

```
In [28]: project_data['teacher_prefix'].value_counts()
```

```
Out[28]: Mrs      57269
Ms        38955
Mr        10648
Teacher   2360
Dr         13
Name: teacher_prefix, dtype: int64
```

```
In [29]: # https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'})
price_data.head(2)
```

```
Out[29]:
```

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

```
In [30]: # join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

```
In [31]: project_data['price'].isnull().any()
```

```
Out[31]: False
```

```
In [32]: #Removing '-' from teacher prefix(as a process of text preprocessing)
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-', '')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-', '')
```

Preparing data for models

```
In [33]: project_data.columns
```

```
Out[33]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
               'project_submitted_datetime', 'project_grade_category', 'project_title',
               'project_resource_summary',
               'teacher_number_of_previously_posted_projects', 'clean_categories',
               'clean_subcategories', 'essay', 'price', 'quantity'],
              dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data (optional)
- quantity : numerical (optional)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

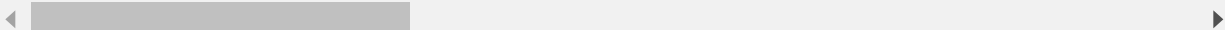
```
In [34]: project_data['text'] = pd.DataFrame(preprocessed_essays)
```

```
In [35]: final_features = ['school_state', 'clean_categories', 'clean_subcategories', 'te
```


In [36]: `project_data.head()`

Out[36]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr	FL	20
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms	AZ	20
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs	KY	20
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs	TX	20



In [37]: `project_data.columns`

Out[37]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state', 'project_submitted_datetime', 'project_grade_category', 'project_title', 'project_resource_summary', 'teacher_number_of_previously_posted_projects', 'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity', 'text'], dtype='object')

In [38]: `X = project_data.copy()`

In [39]: `# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, strati-`

Vectorizing Categorical data

```
In [40]: # Creating empty list to store all the features after vectorization
l = []
```

Encoding categorical features: school_state

```
In [41]: # we use count vectorizer to convert the values into one
vectorizer = CountVectorizer()
vectorizer.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 = vectorizer.transform(X_train['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l.extend(vectorizer.get_feature_names())
```

After vectorizations

(73194, 51) (73194,)

(36051, 51) (36051,)

['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']

=====
=====

Encoding categorical features: teacher_prefix

```
In [42]: X_test['teacher_prefix'].isna().any()
```

Out[42]: False

```
In [43]: vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_prefix'].values).astype(int)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l.extend(vectorizer.get_feature_names())
```

After vectorizations

(73194, 5) (73194,)

(36051, 5) (36051,)

['dr', 'mr', 'mrs', 'ms', 'teacher']

=====

=====

Encoding categorical features: project_grade_category

```
In [44]: vectorizer = CountVectorizer()
vectorizer.fit(X_train['project_grade_category'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['project_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade_category'].values).astype(int)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l.extend(vectorizer.get_feature_names())
```

After vectorizations

(73194, 4) (73194,)

(36051, 4) (36051,)

['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']

=====

=====

Encoding categorical features: clean_subcategories

```
In [45]: trvectorizer = CountVectorizer( lowercase=False, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train

# we use the fitted CountVectorizer to convert the text to vector
X_train_subcat_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_test_subcat_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_subcat_ohe.shape, y_train.shape)
print(X_test_subcat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(73194, 30) (73194,)
(36051, 30) (36051,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government',
'college_careerprep', 'communityservice', 'earlydevelopment', 'economics', 'env
ironmentalscience', 'esl', 'extracurricular', 'financialliteracy', 'foreignlang
uages', 'gym_fitness', 'health_lifescience', 'health_wellness', 'history_geogra
phy', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutritioneduca
tion', 'other', 'parentinvolvement', 'performingarts', 'socialsciences', 'speci
alneeds', 'teamsports', 'visualarts', 'warmth']
=====
=====
```

Encoding categorical features: clean_categories

```
In [46]: vectorizer = CountVectorizer( lowercase=False, binary=True)
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train

# we use the fitted CountVectorizer to convert the text to vector
X_train_cat_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_test_cat_ohe = vectorizer.transform(X_test['clean_categories'].values)

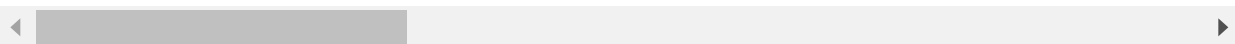
print("After vectorizations")
print(X_train_cat_ohe.shape, y_train.shape)
print(X_test_cat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(73194, 9) (73194,)
(36051, 9) (36051,)
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy
_Language', 'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
=====
=====
```

```
In [47]: X_train.head()
```

Out[47]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proj
103850	119396	p052448	ceec666dc7642501e29f8aaaa39073f8	Mrs	NC	
63919	44075	p179815	2f6270ee9dfca718664671dcc6bc4b49	Ms	CA	
40108	76299	p134030	6213aff3daa8ef576982efc9cc51c388	Mrs	NJ	
65506	21122	p033677	75908c87611ca985b0ce9504048ddd10	Mrs	IL	
43059	64137	p171168	0226c46d1efb9c8d74f6b624475fd518	Mrs	MS	



Encoding numerical features: Price

```
In [48]: from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(X_train['price'].values.reshape(1,-1))

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(1,-1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(1,-1))

X_train_price_norm = X_train_price_norm.reshape(-1,1)
X_test_price_norm = X_test_price_norm.reshape(-1,1)

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
l.extend('price')
```

After vectorizations

(73194, 1) (73194,)

(36051, 1) (36051,)

=====

Encoding numerical features: teacher_number_of_projects

```
In [49]: from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.re

X_train_teacher_number_of_previously_posted_projects_norm = normalizer.transform
X_test_teacher_number_of_previously_posted_projects_norm = normalizer.transform(

X_train_teacher_number_of_previously_posted_projects_norm = X_train_teacher_numbe
X_test_teacher_number_of_previously_posted_projects_norm = X_test_teacher_number

print("After vectorizations")
print(X_train_teacher_number_of_previously_posted_projects_norm.shape, y_train.s
print(X_test_teacher_number_of_previously_posted_projects_norm.shape, y_test.sha
print("="*100)
l.extend('teacher_number_of_previously_posted_projects')
```

After vectorizations

(73194, 1) (73194,)

(36051, 1) (36051,)

=====

Vectorizing Text data

```
In [50]: # Creating separate List for BOW and tfidf
l_BOW = l.copy()
l_tfidf = l.copy()
```

Bag of words

```
In [51]: vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(X_train['text'].values) # fit has to happen only on train data

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

print("After vectorizations")
print(X_train_text_bow.shape, y_train.shape)
print(X_test_text_bow.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l_BOW.extend(vectorizer.get_feature_names())
```

t', 'awaiting', 'awaits', 'awake', 'awaken', 'awakening', 'award', 'awarded', 'awards', 'aware', 'awareness', 'away', 'awe', 'awesome', 'awesomeness', 'awful', 'awhile', 'awkward', 'axles', 'az', 'babes', 'babies', 'baby', 'babyish', 'babysit', 'babysitting', 'baccalaureate', 'back', 'backbone', 'backdrop', 'backdrops', 'backed', 'background', 'backgrounds', 'backing', 'backless', 'backpack', 'backpacks', 'backpatter', 'backs', 'backseat', 'backstage', 'backup', 'backward', 'backwards', 'backyard', 'backyards', 'bacteria', 'bad', 'badge', 'badges', 'badly', 'badminton', 'bag', 'baggage', 'baggie', 'baggies', 'bags', 'bake', 'baked', 'baker', 'bakery', 'baking', 'balance', 'balanced', 'balances', 'balancing', 'ball', 'ballet', 'balloon', 'balloons', 'balls', 'balm', 'balsa', 'baltimore', 'bamboo', 'banana', 'bananas', 'band', 'bands', 'bang', 'banging', 'bangladesh', 'bank', 'bankers', 'banking', 'banks', 'banner', 'banners', 'bar', 'barack', 'barbara', 'bare', 'barely', 'baritone', 'bark', 'barn', 'barnes', 'barred', 'barrel', 'barrels', 'barren', 'barrier', 'barriers', 'bars', 'basal', 'base', 'baseball', 'baseballs', 'based', 'baseline', 'basement', 'baseplates', 'bases', 'basic', 'basically', 'basics', 'basil', 'basis', 'basket', 'basketball', 'basketballs', 'baskets', 'bass', 'basses', 'bat', 'batch', 'bath', 'bathing', 'bathroom', 'bathrooms', 'batman', 'baton', 'batons', 'bats', 'battered', 'batteries', 'battery', 'battin', 'battled', 'battler', 'battling', 'bay', 'beach', 'beaches', 'beacon', 'be

```
In [52]: vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(X_train['project_title'].values.astype('U')) # fit has to happen on training data

# we use the fitted CountVectorizer to convert the text to vector

X_train_title_bow = vectorizer.transform(X_train['project_title'].values.astype('U'))
X_test_title_bow = vectorizer.transform(X_test['project_title'].values.astype('U'))

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_test_title_bow.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l_BOW.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(73194, 2632) (73194,)
(36051, 2632) (36051,)
```

TFIDF vectorizer


```
In [53]: from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['text'].values) # fit has to happen only on train data

# we use the fitted tfidfVectorizer to convert the text to vector
X_train_text_tfidf = vectorizer.transform(X_train['text'].values)
X_test_text_tfidf = vectorizer.transform(X_test['text'].values)

print("After vectorizations")
print(X_train_text_tfidf.shape, y_train.shape)
print(X_test_text_tfidf.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l_tfidf.extend(vectorizer.get_feature_names())
```

```
['00', '000', '10', '100', '1000', '100th', '101', '102', '103', '104', '10
5', '107', '108', '10th', '11', '110', '1100', '115', '11th', '12', '120', '1
200', '123', '125', '12th', '13', '130', '1300', '135', '13th', '14', '140',
'1400', '145', '14th', '15', '150', '1500', '15th', '16', '160', '1600', '16
5', '17', '170', '1700', '175', '17th', '18', '180', '1800', '19', '1950', '1
950s', '1960', '1960s', '1970', '1980s', '1990', '1993', '1999', '19th', '1s
t', '20', '200', '2000', '2001', '2002', '2003', '2004', '2005', '2006', '200
7', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016',
'2017', '2018', '2020', '2025', '2029', '20th', '21', '210', '21st', '22', '2
20', '225', '23', '24', '240', '25', '250', '2500', '26', '260', '27', '270',
'28', '280', '29', '2d', '2nd', '30', '300', '3000', '30am', '31', '3120', '3
2', '320', '33', '330', '34', '340', '35', '350', '36', '360', '365', '37',
'375', '38', '380', '39', '3d', '3doodle', '3doodler', '3doodlers', '3rd', '4
0', '400', '41', '42', '420', '425', '43', '430', '44', '440', '45', '450',
'451', '46', '460', '47', '475', '48', '480', '49', '4c', '4cs', '4k', '4th',
'50', '500', '504', '504s', '51', '52', '520', '53', '530', '54', '540', '5
5', '550', '56', '560', '57', '570', '58', '580', '59', '5k', '5th', '60', '6
00', '61', '62', '63', '630', '64', '65', '650', '66', '67', '68', '69', '6p
'
```

```
In [54]: from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['project_title'].values.astype('U')) # fit has to happen on training data

# we use the fitted tfidfVectorizer to convert the text to vector
X_train_title_tfidf = vectorizer.transform(X_train['project_title'].values.astype('U'))
X_test_title_tfidf = vectorizer.transform(X_test['project_title'].values.astype('U'))

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
l_tfidf.extend(vectorizer.get_feature_names())
```

```
After vectorizations
(73194, 2632) (73194,)
(36051, 2632) (36051,)
```

1.5.4 Merging all the above features

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

```
In [55]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((X_train_state_ohe,X_train_teacher_ohe,X_train_grade_ohe,X_train_c
X_te = hstack((X_test_state_ohe,X_test_teacher_ohe,X_test_grade_ohe,X_test_cat_o
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
print("=="*100)
```

Final Data matrix

(73194, 16850) (73194,)

(36051, 16850) (36051,)

=====
=====

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 paramter tuning(find best Alpha)

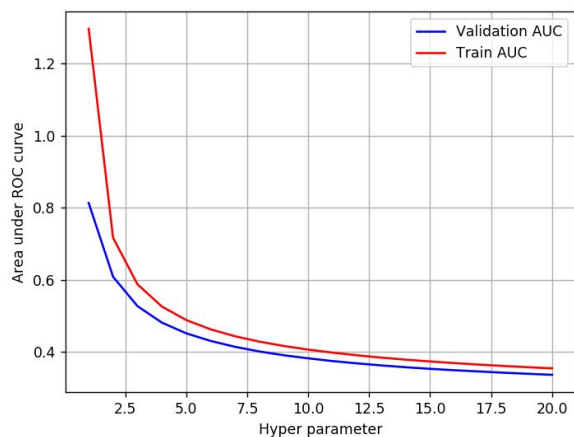
- Find the best hyper parameter which will give the maximum [AUC](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
- Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
- Find the best hyper paramter 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

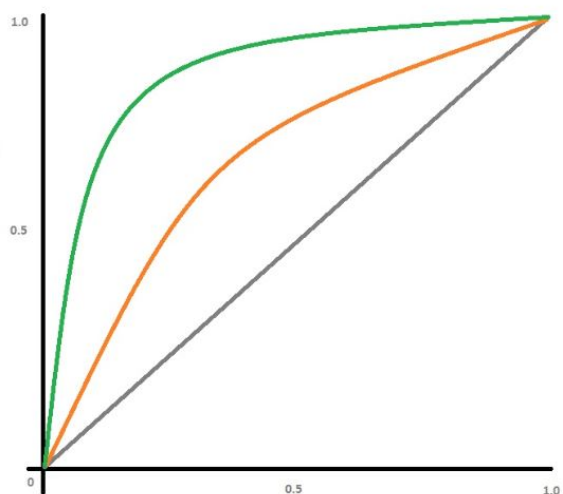
- Find the top 10 features of positive class and top 10 features of negative class for both feature sets **Set 1** and **Set 2** using values of `feature_log_prob_` parameter of [MultinomialNB](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.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.appliedaia.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) (<https://www.appliedaia.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](#).

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

(<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/) (<http://zetcode.com/python/prettytable/>)

Vectorizer	Model	Hyper parameter	AUC
BOW	Brute	7	0.78
TFIDF	Brute	12	0.79
W2V	Brute	10	0.78
TFIDFW2V	Brute	6	0.78

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. \(https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf\)](https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

Naive Bayes

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 instructions

Applying Naive Bayes on BOW, SET 1

```
In [56]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV
import warnings
import math
warnings.filterwarnings("ignore")
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(fit_prior=False)
parameters = {'alpha':[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]}
clf = GridSearchCV(nb, parameters, cv=10, scoring='roc_auc', n_jobs=-1)
clf.fit(X_tr, 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']

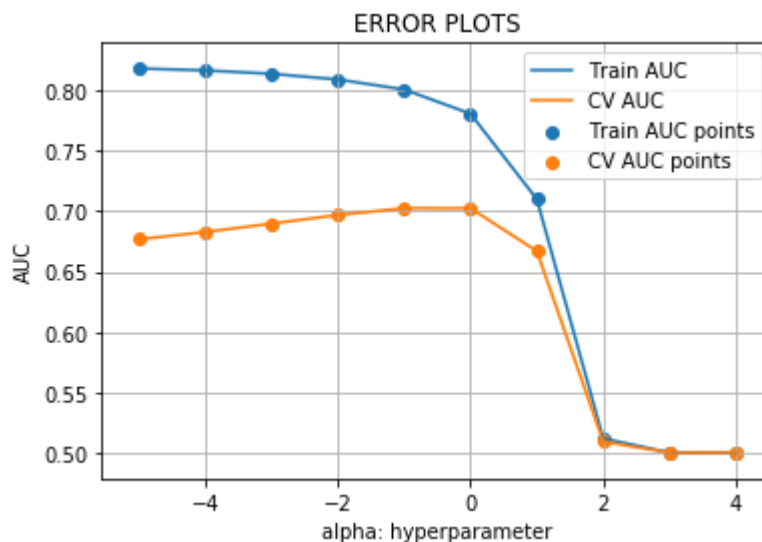
parameters['alpha'] = [math.log10(i) for i in parameters['alpha']]

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)

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)

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()
```



In [57]: `print(cv_auc)`

```
[0.67679396 0.68277996 0.68977703 0.69686659 0.70254857 0.7025039
 0.66703615 0.50914662 0.5          0.50004046]
```

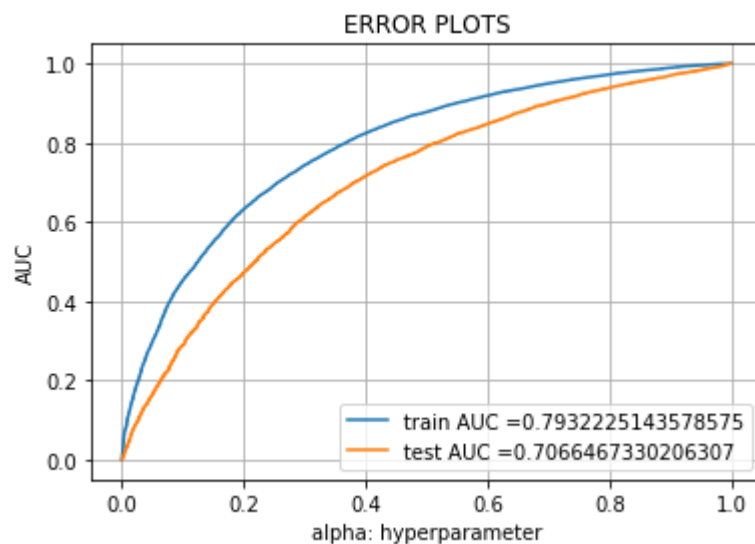
```
In [58]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
from sklearn.metrics import roc_curve, auc
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha=0.1, class_prior=[0.5, 0.5])
nb.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs

y_train_pred = nb.predict_proba(X_tr)[: , 1]
y_test_pred = nb.predict_proba(X_te)[: , 1]

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 [59]: #we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba,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

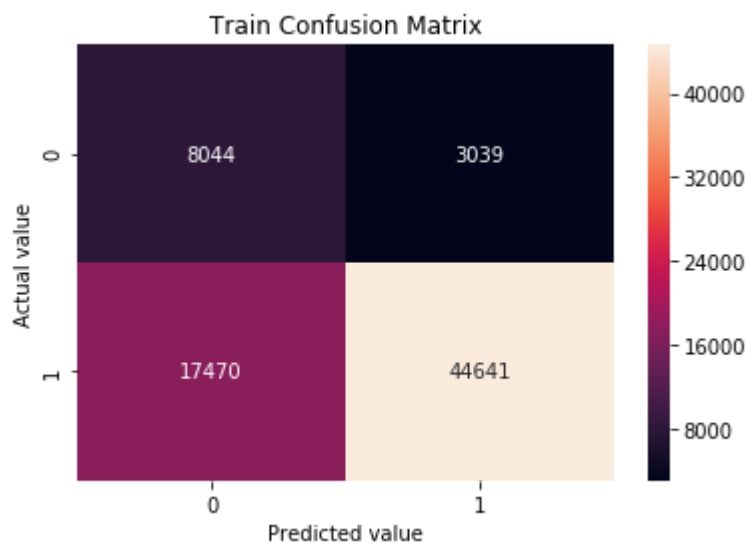
    print("the maximum value of tpr*(1-fpr)",max(tpr*(1-fpr)))
    print("Threshold:",np.round(t,3))
    prediction=[]
    for i in proba:
        if i>=t:
            prediction.append(1)
        else:
            prediction.append(0)

    return prediction
```

```
In [60]: import seaborn as sns
#https://stackoverflow.com/a/33158941/10967428
con_tr=confusion_matrix(y_train,predict(y_train_pred,tr_thresholds,train_fpr,tra:

sns.heatmap(con_tr,annot=True,fmt='0.00f',annot_kws={'size':10})
plt.title("Train Confusion Matrix")
plt.ylabel("Actual value")
plt.xlabel("Predicted value")
plt.show()
```

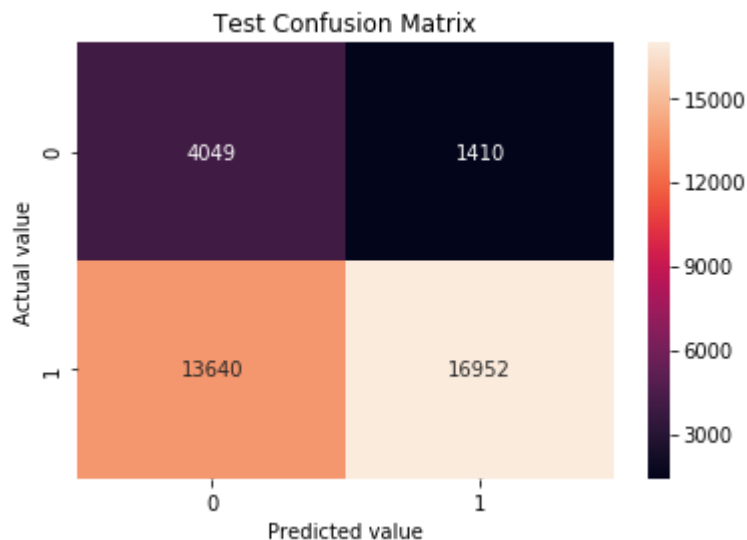
the maximum value of $tpr*(1-fpr)$ 0.5216510931356078
 Threshold: 0.515




```
In [61]: #https://stackoverflow.com/a/33158941/10967428
con_te=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, te:

sns.heatmap(con_te,annot=True,fmt='0.00f',annot_kws={'size':10})
plt.title("Test Confusion Matrix")
plt.ylabel("Actual value")
plt.xlabel("Predicted value")
plt.show()
```

the maximum value of $tpr*(1-fpr)$ 0.4355982591988509
Threshold: 0.911



Top 10 important features of positive class from SET 1

```
In [62]: pos_class_prob_sorted = nb.feature_log_prob_[1].argsort()
print("The top 10 positive features are")
print(np.take(l_BOW, pos_class_prob_sorted[-10:]))
```

The top 10 positive features are
['narrowed' 'mummies' 'mallet' 'headsets' 'laughs' 'nine' 'circulating'
'launcher' 'says' 'stressed']

Top 10 important features of negative class from SET 1

```
In [63]: neg_class_prob_sorted = nb.feature_log_prob_[0].argsort()
print("The top 10 negative features are")
print(np.take(l_BOW, neg_class_prob_sorted[-10:]))
```

The top 10 negative features are
['narrowed' 'mallet' 'mummies' 'headsets' 'laughs' 'nine' 'circulating'
'launcher' 'says' 'stressed']

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [64]: *Write all the code with proper documentation*
sparse matrices: <https://stackoverflow.com/a/19710648/4084039>

```

sparse import hstack
hstack((X_train_state_ohe,X_train_teacher_ohe,X_train_grade_ohe,X_train_subcat_ohe,X_train_
hstack((X_test_state_ohe,X_test_teacher_ohe,X_test_grade_ohe,X_test_subcat_ohe,X_test_
Final Data matrix")
X_train.shape, y_train.shape)
X_test.shape, y_test.shape)
print(X_train.shape)

```

Final Data matrix

(73194, 16841) (73194,)

(36051, 16841) (36051,)

=====

```
In [65]: # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV
import warnings
import math
warnings.filterwarnings("ignore")
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(fit_prior=False)
parameters = {'alpha':[0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]}
clf = GridSearchCV(nb, parameters, cv=10, scoring='roc_auc', n_jobs=-1)
clf.fit(X_tr, 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']

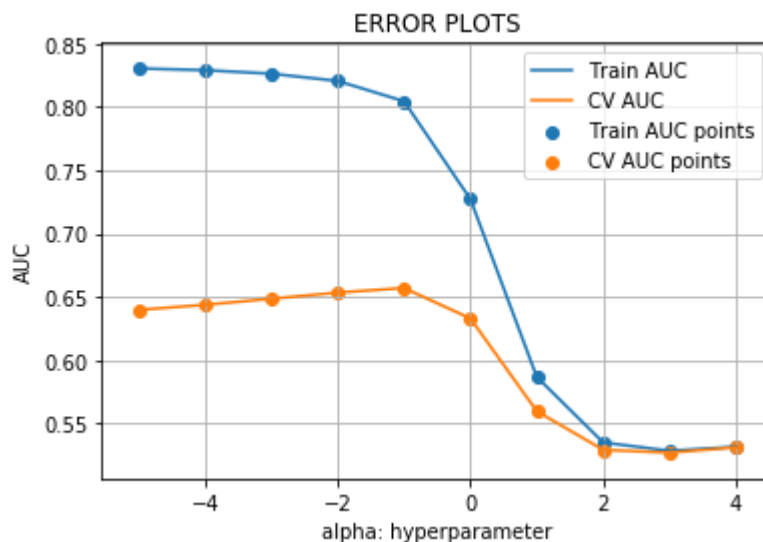
parameters['alpha'] = [math.log10(i) for i in parameters['alpha']]

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)

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)

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()
```



In [66]: `print(cv_auc)`

```
[0.63975429 0.64377672 0.64859664 0.65341323 0.65702367 0.63299836
 0.55974186 0.5292641  0.52704006 0.53125385]
```

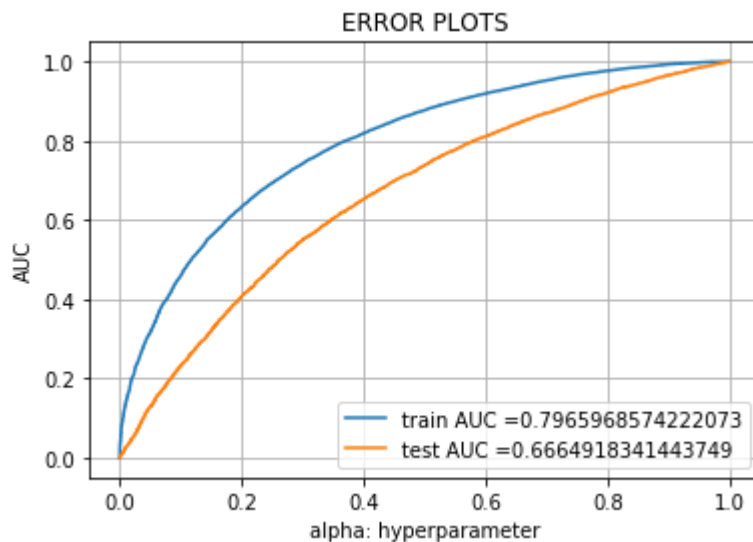
```
In [67]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
from sklearn.metrics import roc_curve, auc
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB(alpha=0.1, class_prior=[0.5, 0.5])
nb.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs

y_train_pred = nb.predict_proba(X_tr)[: , 1]
y_test_pred = nb.predict_proba(X_te)[: , 1]

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 [68]: #we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba,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

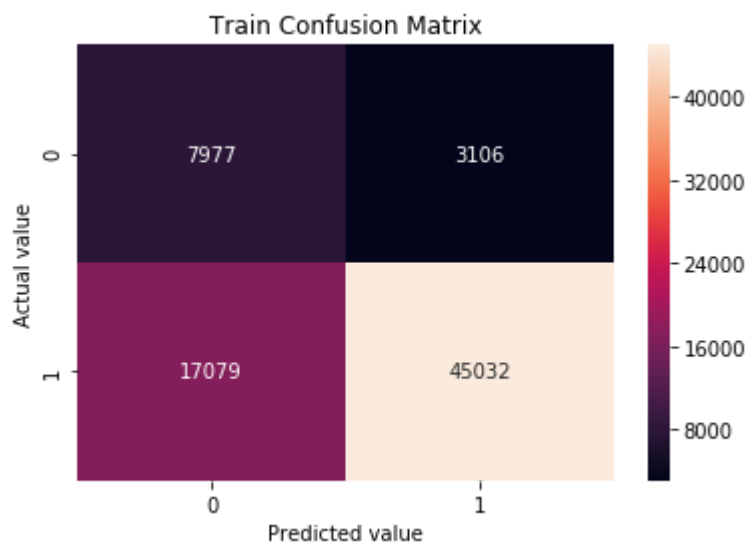
    print("the maximum value of tpr*(1-fpr)",max(tpr*(1-fpr)))
    print("Threshold:",np.round(t,3))
    prediction=[]
    for i in proba:
        if i>=t:
            prediction.append(1)
        else:
            prediction.append(0)

    return prediction
```

```
In [69]: import seaborn as sns
#https://stackoverflow.com/a/33158941/10967428
con_tr=confusion_matrix(y_train,predict(y_train_pred,tr_thresholds,train_fpr,tra:

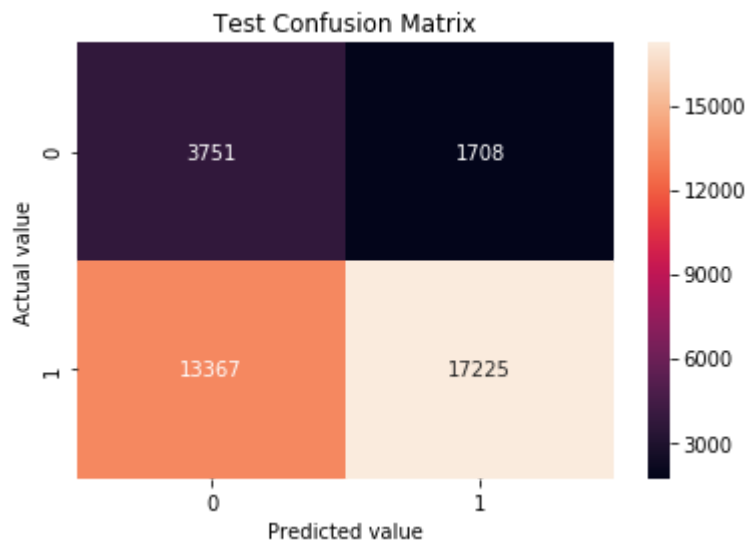
sns.heatmap(con_tr,annot=True,fmt='0.00f',annot_kws={'size':10})
plt.title("Train Confusion Matrix")
plt.ylabel("Actual value")
plt.xlabel("Predicted value")
plt.show()
```

the maximum value of $tpr*(1-fpr)$ 0.5218371251302956
 Threshold: 0.498



```
In [70]: #https://stackoverflow.com/a/33158941/10967428
con_te=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, te
sns.heatmap(con_te,annot=True,fmt='0.00f',annot_kws={'size':10})
plt.title("Test Confusion Matrix")
plt.ylabel("Actual value")
plt.xlabel("Predicted value")
plt.show()
```

the maximum value of $tpr*(1-fpr)$ 0.39404258140370857
Threshold: 0.564



Top 10 important features of positive class from SET 1

```
In [71]: pos_class_prob_sorted = nb.feature_log_prob_[1].argsort()
print("The top 10 positive features are")
print(np.take(l_BOW, pos_class_prob_sorted[-10:]))
```

The top 10 positive features are
['streets' 'ca' 'grades_6_8' 'literature_writing' 'mathematics' 'literacy'
'grades_3_5' 'ms' 'grades_prek_2' 'mrs']

Top 10 important features of negative class from SET 1

```
In [72]: neg_class_prob_sorted = nb.feature_log_prob_[0].argsort()
print("The top 10 negative features are")
print(np.take(l_BOW, neg_class_prob_sorted[-10:]))
```

The top 10 negative features are
['ca' 'specialneeds' 'grades_6_8' 'literature_writing' 'literacy'
'mathematics' 'grades_3_5' 'ms' 'grades_prek_2' 'mrs']

Conclusions

```
In [73]: # Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Hyper parameter", "AUC"]

x.add_row(["BOW", 0.1, 0.7])
x.add_row(["TFIDF", 0.1, 0.66])

print(x)
```

```
+-----+-----+-----+
| Vectorizer | Hyper parameter | AUC |
+-----+-----+-----+
|      BOW   |           0.1    | 0.7 |
|      TFIDF |           0.1    | 0.66|
+-----+-----+-----+
```

Summary

1. On comparing both the results we see that TFIDF featurization works a bit well in terms precision and recall
2. Its very good compared to kNN in terms of execution time and in terms of accuracy