DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example**
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targete enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories f following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal ((https://en.wikipedia.org/wiki/List_of_U.S._state_abbr **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategori **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the projec • `My students need hands on literacy mater sensory needs!`

Feature	Description
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Ex** 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed pro bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated value • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted b **Example:** 2

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [6]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

In [5]:

```
!pip install plotly
```

```
Collecting plotly
  Downloading https://files.pythonhosted.org/packages/ff/75/3982bac5076d0c
e6d23103c03840fcaec90c533409f9d82c19f54512a38a/plotly-3.10.0-py2.py3-none-
any.whl (41.5MB)
Requirement already satisfied: pytz in c:\programdata\anaconda3\lib\site-p
ackages (from plotly) (2018.4)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-pa
ckages (from plotly) (1.11.0)
Collecting retrying>=1.3.3 (from plotly)
  Downloading https://files.pythonhosted.org/packages/44/ef/beae4b4ef80902
f22e3af073397f079c96969c69b2c7d52a57ea9ae61c9d/retrying-1.3.3.tar.gz
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\si
te-packages (from plotly) (2.18.4)
Requirement already satisfied: decorator>=4.0.6 in c:\programdata\anaconda
3\lib\site-packages (from plotly) (4.3.0)
Requirement already satisfied: nbformat>=4.2 in c:\programdata\anaconda3\l
ib\site-packages (from plotly) (4.4.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\programdata\ana
conda3\lib\site-packages (from requests->plotly) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in c:\programdata\anaconda3
\lib\site-packages (from requests->plotly) (2.6)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in c:\programdata\ana
conda3\lib\site-packages (from requests->plotly) (1.22)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anacon
da3\lib\site-packages (from requests->plotly) (2018.11.29)
Requirement already satisfied: ipython_genutils in c:\programdata\anaconda
3\lib\site-packages (from nbformat>=4.2->plotly) (0.2.0)
Requirement already satisfied: traitlets>=4.1 in c:\programdata\anaconda3
\lib\site-packages (from nbformat>=4.2->plotly) (4.3.2)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\programdata\a
naconda3\lib\site-packages (from nbformat>=4.2->plotly) (2.6.0)
Requirement already satisfied: jupyter_core in c:\programdata\anaconda3\li
b\site-packages (from nbformat>=4.2->plotly) (4.4.0)
Building wheels for collected packages: retrying
  Running setup.py bdist_wheel for retrying: started
  Running setup.py bdist_wheel for retrying: finished with status 'done'
  Stored in directory: C:\Users\Prof Arkopal Goswami\AppData\Local\pip\Cac
he\wheels\d7\a9\33\acc7b709e2a35caa7d4cae442f6fe6fbf2c43f80823d46460c
Successfully built retrying
Installing collected packages: retrying, plotly
Successfully installed plotly-3.10.0 retrying-1.3.3


wxpython 4.0.3 requires PyPubSub, which is not installed.
distributed 1.21.8 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pi
p' command.
```

In [4]:

```
!pip install gensim
```

```
Collecting gensim
  Downloading https://files.pythonhosted.org/packages/81/80/858ef502e80baa
6384b75fd5c89f01074b791a13b830487f9e25bdce50ec/gensim-3.7.3.tar.gz (23.4M
B)
Requirement already satisfied: numpy>=1.11.3 in c:\programdata\anaconda3\l
ib\site-packages (from gensim) (1.14.3)
Requirement already satisfied: scipy>=0.18.1 in c:\programdata\anaconda3\l
ib\site-packages (from gensim) (1.1.0)
Requirement already satisfied: six>=1.5.0 in c:\programdata\anaconda3\lib
\site-packages (from gensim) (1.11.0)
Collecting smart_open>=1.7.0 (from gensim)
  Downloading https://files.pythonhosted.org/packages/37/c0/25d19badc49542
8dec6a4bf7782de617ee0246a9211af75b302a2681dea7/smart_open-1.8.4.tar.gz (63
kB)
Requirement already satisfied: boto>=2.32 in c:\programdata\anaconda3\lib
\site-packages (from smart_open>=1.7.0->gensim) (2.48.0)
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\si
te-packages (from smart_open>=1.7.0->gensim) (2.18.4)
Collecting boto3 (from smart_open>=1.7.0->gensim)
  Downloading https://files.pythonhosted.org/packages/3b/6e/8189e561e15eb0
def4f3155ab4addf231891b2df2216495fc58a5049af55/boto3-1.9.180-py2.py3-none-
any.whl (128kB)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in c:\programdata\ana
conda3\lib\site-packages (from requests->smart_open>=1.7.0->gensim) (3.0.
4)
Requirement already satisfied: idna<2.7,>=2.5 in c:\programdata\anaconda3
\lib\site-packages (from requests->smart_open>=1.7.0->gensim) (2.6)
Requirement already satisfied: urllib3<1.23,>=1.21.1 in c:\programdata\ana
conda3\lib\site-packages (from requests->smart_open>=1.7.0->gensim) (1.22)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anacon
da3\lib\site-packages (from requests->smart_open>=1.7.0->gensim) (2018.11.
29)
Collecting botocore<1.13.0,>=1.12.180 (from boto3->smart_open>=1.7.0->gens
im)
  Downloading https://files.pythonhosted.org/packages/3b/27/fa7da6feb20d1d
fc0ab562226061b20da2d27ea18ca32dc764fe86704a99/botocore-1.12.180-py2.py3-n
one-any.whl (5.6MB)
Collecting s3transfer<0.3.0,>=0.2.0 (from boto3->smart_open>=1.7.0->gensi
m)
  Downloading https://files.pythonhosted.org/packages/16/8a/1fc3dba0c4923c
2a76e1ff0d52b305c44606da63f718d14d3231e21c51b0/s3transfer-0.2.1-py2.py3-no
ne-any.whl (70kB)
Collecting jmespath<1.0.0,>=0.7.1 (from boto3->smart_open>=1.7.0->gensim)
  Downloading https://files.pythonhosted.org/packages/83/94/7179c3832a6d45
b266ddb2aac329e101367fbdb11f425f13771d27f225bb/jmespath-0.9.4-py2.py3-none
-any.whl
Requirement already satisfied: python-dateutil<3.0.0,>=2.1; python_version
>= "2.7" in c:\programdata\anaconda3\lib\site-packages (from botocore<1.1
3.0,>=1.12.180->boto3->smart_open>=1.7.0->gensim) (2.7.3)
Requirement already satisfied: docutils>=0.10 in c:\programdata\anaconda3
\lib\site-packages (from botocore<1.13.0,>=1.12.180->boto3->smart_open>=1.
7.0->gensim) (0.14)
Building wheels for collected packages: gensim, smart-open
  Running setup.py bdist_wheel for gensim: started
  Running setup.py bdist_wheel for gensim: finished with status 'done'
  Stored in directory: C:\Users\Prof Arkopal Goswami\AppData\Local\pip\Cac
he\wheels\73\6b\89\bb14fd56b74774a39a771a12f525a6a14c2c2692d3084ad048
  Running setup.py bdist_wheel for smart-open: started
  Running setup.py bdist_wheel for smart-open: finished with status 'done'
  Stored in directory: C:\Users\Prof Arkopal Goswami\AppData\Local\pip\Cac
he\wheels\5f\ea\fb\5b1a947b369724063b2617011f1540c44eb00e28c3d2ca8692
```

```
Successfully built gensim smart-open
Installing collected packages: jmespath, botocore, s3transfer, boto3, smar
t-open, gensim
Successfully installed boto3-1.9.180 botocore-1.12.180 gensim-3.7.3 jmespa
th-0.9.4 s3transfer-0.2.1 smart-open-1.8.4



wxpython 4.0.3 requires PyPubSub, which is not installed.
distributed 1.21.8 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pi
p' command.
```

In [2]:

```
!pip install tqdm
```

```
Collecting tqdm
  Downloading https://files.pythonhosted.org/packages/9f/3d/7a6b68b631d2ab
54975f3a4863f3c4e9b26445353264ef01f465dc9b0208/tqdm-4.32.2-py2.py3-none-an
y.whl (50kB)
Installing collected packages: tqdm
Successfully installed tqdm-4.32.2

wxpython 4.0.3 requires PyPubSub, which is not installed.
distributed 1.21.8 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pi
p' command.
```

1.1 Reading Data

In [7]:

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

In [8]:

```
project_data.shape
```

Out[8]:

```
(109248, 17)
```

In [9]:

```
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 [10]:

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

Out[10]:

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

In [11]:

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


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
39
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


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


project_data.head(2)
```

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	scho
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

Adding resource data in dataframe

In [12]:

```
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[12]:

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

In [13]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_i
ndex()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [14]:

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

Out[14]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_st
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

In [15]:

```
project_data.shape
```

Out[15]:

```
(109248, 19)
```

In [16]:

```
project_data.columns
```

Out[16]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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_approve
d',
       'price', 'quantity'],
      dtype='object')
```

1.2 preprocessing of `project_subject_categories`

In [17]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        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 [18]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        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/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

In [19]:

```
project_data.columns
```

Out[19]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', '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_approve
d',
       'price', 'quantity', 'clean_categories', 'clean_subcategories'],
      dtype='object')
```

1.3 Text preprocessing

In [20]:

```
# 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 [21]:

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

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_st
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

In [22]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [23]:

```
# 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[99999])
print("="*50)
```

I have been fortunate enough to use the Fairy Tale STEM kits in my classro
om as well as the STEM journals, which my students really enjoyed. I woul
d love to implement more of the Lakeshore STEM kits in my classroom for th
e next school year as they provide excellent and engaging STEM lessons.My
students come from a variety of backgrounds, including language and socioe
conomic status. Many of them don't have a lot of experience in science an
d engineering and these kits give me the materials to provide these exciti
ng opportunities for my students.Each month I try to do several science or
STEM/STEAM projects. I would use the kits and robot to help guide my scie
nce instruction in engaging and meaningful ways. I can adapt the kits to
my current language arts pacing guide where we already teach some of the m
aterial in the kits like tall tales (Paul Bunyan) or Johnny Appleseed. Th
e following units will be taught in the next school year where I will impl
ement these kits: magnets, motion, sink vs. float, robots. I often get to
these units and don't know If I am teaching the right way or using the rig
ht materials. The kits will give me additional ideas, strategies, and l
essons to prepare my students in science.It is challenging to develop high
quality science activities. These kits give me the materials I need to pr
ovide my students with science activities that will go along with the curr
iculum in my classroom. Although I have some things (like magnets) in my
classroom, I don't know how to use them effectively. The kits will provid
e me with the right amount of materials and show me how to use them in an
appropriate way.

==========

I teach high school English to students with learning and behavioral disab
ilities. My students all vary in their ability level. However, the ultimat
e goal is to increase all students literacy levels. This includes their re
ading, writing, and communication levels.I teach a really dynamic group of
students. However, my students face a lot of challenges. My students all l
ive in poverty and in a dangerous neighborhood. Despite these challenges,
I have students who have the the desire to defeat these challenges. My stu
dents all have learning disabilities and currently all are performing belo
w grade level. My students are visual learners and will benefit from a cla
ssroom that fulfills their preferred learning style.The materials I am req
uesting will allow my students to be prepared for the classroom with the n
ecessary supplies. Too often I am challenged with students who come to sc
hool unprepared for class due to economic challenges. I want my students
to be able to focus on learning and not how they will be able to get schoo
l supplies. The supplies will last all year. Students will be able to co
mplete written assignments and maintain a classroom journal. The chart pa
per will be used to make learning more visual in class and to create poste
rs to aid students in their learning. The students have access to a class
room printer. The toner will be used to print student work that is comple
ted on the classroom Chromebooks.I want to try and remove all barriers for
the students learning and create opportunities for learning. One of the bi
ggest barriers is the students not having the resources to get pens, pape
r, and folders. My students will be able to increase their literacy skills
because of this project.

==========

\"Life moves pretty fast. If you don't stop and look around once in awhil
e, you could miss it.\" from the movie, Ferris Bueller's Day Off. Think
back...what do you remember about your grandparents? How amazing would it
be to be able to flip through a book to see a day in their lives?My second
graders are voracious readers! They love to read both fiction and nonficti
on books. Their favorite characters include Pete the Cat, Fly Guy, Piggie
and Elephant, and Mercy Watson. They also love to read about insects, spac
e and plants. My students are hungry bookworms! My students are eager to l
earn and read about the world around them. My kids love to be at school an
d are like little sponges absorbing everything around them. Their parents
work long hours and usually do not see their children. My students are usu
ally cared for by their grandparents or a family friend. Most of my studen

ts do not have someone who speaks English at home. Thus it is difficult fo r my students to acquire language.Now think forward... wouldn't it mean a lot to your kids, nieces or nephews or grandchildren, to be able to see a day in your life today 30 years from now? Memories are so precious to us a nd being able to share these memories with future generations will be a re warding experience. As part of our social studies curriculum, students wi ll be learning about changes over time. Students will be studying photos to learn about how their community has changed over time. In particular, we will look at photos to study how the land, buildings, clothing, and sch ools have changed over time. As a culminating activity, my students will capture a slice of their history and preserve it through scrap booking. Ke y important events in their young lives will be documented with the date, location, and names. Students will be using photos from home and from sc hool to create their second grade memories. Their scrap books will prese rve their unique stories for future generations to enjoy.Your donation to this project will provide my second graders with an opportunity to learn a bout social studies in a fun and creative manner. Through their scrapbook s, children will share their story with others and have a historical docum ent for the rest of their lives.

==========

\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the small est students with the biggest enthusiasm for learning. My students learn i n many different ways using all of our senses and multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nStud ents in my class come from a variety of different backgrounds which makes for wonderful sharing of experiences and cultures, including Native Americ ans.\r\nOur school is a caring community of successful learners which can be seen through collaborative student project based learning in and out of the classroom. Kindergarteners in my class love to work with hands-on mate rials and have many different opportunities to practice a skill before it is mastered. Having the social skills to work cooperatively with friends i s a crucial aspect of the kindergarten curriculum.Montana is the perfect p lace to learn about agriculture and nutrition. My students love to role pl ay in our pretend kitchen in the early childhood classroom. I have had sev eral kids ask me, \"Can we try cooking with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we learn important m ath and writing concepts while cooking delicious healthy food for snack ti me. My students will have a grounded appreciation for the work that went i nto making the food and knowledge of where the ingredients came from as we ll as how it's healthy for their bodies. This project would expand our lea rning of nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce, make our own bread, and mix up hea lthy plants from our classroom garden in the spring. We will also create o ur own cookbooks to be printed and shared with families. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healt hy cooking.nannan

==========

My classroom consists of twenty-two amazing sixth graders from different c ultures and backgrounds. They are a social bunch who enjoy working in part ners and working with groups. They are hard-working and eager to head to m iddle school next year. My job is to get them ready to make this transitio n and make it as smooth as possible. In order to do this, my students need to come to school every day and feel safe and ready to learn. Because they are getting ready to head to middle school, I give them lots of choice- ch oice on where to sit and work, the order to complete assignments, choice o f projects, etc. Part of the students feeling safe is the ability for them to come into a welcoming, encouraging environment. My room is colorful and the atmosphere is casual. I want them to take ownership of the classroom b ecause we ALL share it together. Because my time with them is limited, I w ant to ensure they get the most of this time and enjoy it to the best of t heir abilities.Currently, we have twenty-two desks of differing sizes, yet

```
the desks are similar to the ones the students will use in middle school.
We also have a kidney table with crates for seating. I allow my students t
o choose their own spots while they are working independently or in group
s. More often than not, most of them move out of their desks and onto the
crates. Believe it or not, this has proven to be more successful than maki
ng them stay at their desks! It is because of this that I am looking towar
d the “Flexible Seating” option for my classroom.\r\n The students look fo
rward to their work time so they can move around the room. I would like to
get rid of the constricting desks and move toward more “fun” seating optio
ns. I am requesting various seating so my students have more options to si
t. Currently, I have a stool and a papasan chair I inherited from the prev
ious sixth-grade teacher as well as five milk crate seats I made, but I wo
uld like to give them more options and reduce the competition for the “goo
d seats”. I am also requesting two rugs as not only more seating options b
ut to make the classroom more welcoming and appealing. In order for my stu
dents to be able to write and complete work without desks, I am requesting
a class set of clipboards. Finally, due to curriculum that requires groups
to work together, I am requesting tables that we can fold up when we are n
ot using them to leave more room for our flexible seating options.\r\nI kn
ow that with more seating options, they will be that much more excited abo
ut coming to school! Thank you for your support in making my classroom one
students will remember forever!nannan
==================================================
```

In [24]:

```
# 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 [25]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smal
lest students with the biggest enthusiasm for learning. My students learn
in many different ways using all of our senses and multiple intelligences.
I use a wide range of techniques to help all my students succeed. \r\nStud
ents in my class come from a variety of different backgrounds which makes
for wonderful sharing of experiences and cultures, including Native Americ
ans.\r\nOur school is a caring community of successful learners which can
be seen through collaborative student project based learning in and out of
the classroom. Kindergarteners in my class love to work with hands-on mate
rials and have many different opportunities to practice a skill before it
is mastered. Having the social skills to work cooperatively with friends i
s a crucial aspect of the kindergarten curriculum.Montana is the perfect p
lace to learn about agriculture and nutrition. My students love to role pl
ay in our pretend kitchen in the early childhood classroom. I have had sev
eral kids ask me, \"Can we try cooking with REAL food?\" I will take their
idea and create \"Common Core Cooking Lessons\" where we learn important m
ath and writing concepts while cooking delicious healthy food for snack ti
me. My students will have a grounded appreciation for the work that went i
nto making the food and knowledge of where the ingredients came from as we
ll as how it is healthy for their bodies. This project would expand our le
arning of nutrition and agricultural cooking recipes by having us peel our
own apples to make homemade applesauce, make our own bread, and mix up hea
lthy plants from our classroom garden in the spring. We will also create o
ur own cookbooks to be printed and shared with families. \r\nStudents will
gain math and literature skills as well as a life long enjoyment for healt
hy cooking.nannan
==================================================
```

In [26]:

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

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smalle
st students with the biggest enthusiasm for learning. My students learn in
many different ways using all of our senses and multiple intelligences. I
use a wide range of techniques to help all my students succeed.   Students
in my class come from a variety of different backgrounds which makes for w
onderful sharing of experiences and cultures, including Native Americans.
Our school is a caring community of successful learners which can be seen
through collaborative student project based learning in and out of the cla
ssroom. Kindergarteners in my class love to work with hands-on materials a
nd have many different opportunities to practice a skill before it is mast
ered. Having the social skills to work cooperatively with friends is a cru
cial aspect of the kindergarten curriculum.Montana is the perfect place to
learn about agriculture and nutrition. My students love to role play in ou
r pretend kitchen in the early childhood classroom. I have had several kid
s ask me,  Can we try cooking with REAL food?  I will take their idea and
create  Common Core Cooking Lessons  where we learn important math and wri
ting concepts while cooking delicious healthy food for snack time. My stud
ents will have a grounded appreciation for the work that went into making
the food and knowledge of where the ingredients came from as well as how i
t is healthy for their bodies. This project would expand our learning of n
utrition and agricultural cooking recipes by having us peel our own apples
to make homemade applesauce, make our own bread, and mix up healthy plants
from our classroom garden in the spring. We will also create our own cookb
ooks to be printed and shared with families.   Students will gain math and
literature skills as well as a life long enjoyment for healthy cooking.nan
nan
```

In [27]:

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

```
 A person is a person no matter how small Dr Seuss I teach the smallest st
udents with the biggest enthusiasm for learning My students learn in many
different ways using all of our senses and multiple intelligences I use a
wide range of techniques to help all my students succeed Students in my cl
ass come from a variety of different backgrounds which makes for wonderful
sharing of experiences and cultures including Native Americans Our school
is a caring community of successful learners which can be seen through col
laborative student project based learning in and out of the classroom Kind
ergarteners in my class love to work with hands on materials and have many
different opportunities to practice a skill before it is mastered Having t
he social skills to work cooperatively with friends is a crucial aspect of
the kindergarten curriculum Montana is the perfect place to learn about ag
riculture and nutrition My students love to role play in our pretend kitch
en in the early childhood classroom I have had several kids ask me Can we
try cooking with REAL food I will take their idea and create Common Core C
ooking Lessons where we learn important math and writing concepts while co
oking delicious healthy food for snack time My students will have a ground
ed appreciation for the work that went into making the food and knowledge
of where the ingredients came from as well as how it is healthy for their
bodies This project would expand our learning of nutrition and agricultura
l cooking recipes by having us peel our own apples to make homemade apples
auce make our own bread and mix up healthy plants from our classroom garde
n in the spring We will also create our own cookbooks to be printed and sh
ared with families Students will gain math and literature skills as well a
s a life long enjoyment for healthy cooking nannan
```

In [28]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [29]:

```
# 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.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████
███| 109248/109248 [00:46<00:00, 2330.36it/s]
```

In [30]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[30]:

```
'person person no matter small dr seuss teach smallest students biggest en
thusiasm learning students learn many different ways using senses multiple
intelligences use wide range techniques help students succeed students cla
ss come variety different backgrounds makes wonderful sharing experiences
cultures including native americans school caring community successful lea
rners seen collaborative student project based learning classroom kinderga
rteners class love work hands materials many different opportunities pract
ice skill mastered social skills work cooperatively friends crucial aspect
kindergarten curriculum montana perfect place learn agriculture nutrition
students love role play pretend kitchen early childhood classroom several
kids ask try cooking real food take idea create common core cooking lesson
s learn important math writing concepts cooking delicious healthy food sna
ck time students grounded appreciation work went making food knowledge ing
redients came well healthy bodies project would expand learning nutrition
agricultural cooking recipes us peel apples make homemade applesauce make
bread mix healthy plants classroom garden spring also create cookbooks pri
nted shared families students gain math literature skills well life long e
njoyment healthy cooking nannan'
```

In [31]:

```
project_data['preprocessed_essays'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
```

In [32]:

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

Out[32]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_st
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

1.4 Preprocessing of `project_title`

In [33]:

```
# similarly you can preprocess the titles also
```

In [34]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_project_title = []
# 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_project_title.append(sent.lower().strip())
```

100%|██████████| 109248/109248 [00:02<00:00, 46720.06it/s]

In [35]:

```
# after preprocesing
preprocessed_project_title[1000]
```

Out[35]:

'empowering students through art learning about then now'

In [36]:

```
#https://stackoverflow.com/questions/26666919/add-column-in-dataframe-from-list/38490727
project_data['preprocessed_project_title'] = preprocessed_project_title
project_data.drop(['project_title'], axis=1, inplace=True)
```

In [37]:

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

Out[37]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_st
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT

In [38]:

```
project_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 109248 entries, 0 to 109247
Data columns (total 20 columns):
Unnamed: 0                                      109248 non-null int64
id                                              109248 non-null object
teacher_id                                      109248 non-null object
teacher_prefix                                  109245 non-null object
school_state                                    109248 non-null object
Date                                            109248 non-null datetime64
[ns]
project_grade_category                          109248 non-null object
project_essay_1                                 109248 non-null object
project_essay_2                                 109248 non-null object
project_essay_3                                 3758 non-null object
project_essay_4                                 3758 non-null object
project_resource_summary                        109248 non-null object
teacher_number_of_previously_posted_projects    109248 non-null int64
project_is_approved                             109248 non-null int64
price                                           109248 non-null float64
quantity                                        109248 non-null int64
clean_categories                                109248 non-null object
clean_subcategories                             109248 non-null object
preprocessed_essays                             109248 non-null object
preprocessed_project_title                      109248 non-null object
dtypes: datetime64[ns](1), float64(1), int64(4), object(14)
memory usage: 17.5+ MB
```

In [39]:

```
#df.drop(df.columns[[0,1,2,5,7,8,9,10,]], axis=1, inplace=True)
```

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

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

5. **Conclusion** (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

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

    - 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 (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

Note: Data Leakage

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

2. Naive Bayes

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

In [40]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

from collections import Counter
from sklearn.metrics import accuracy_score

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
```

In [41]:

```
y=project_data['project_is_approved']
y.shape
```

Out[41]:

```
(109248,)
```

In [42]:

```
#replace NAN to space https ://stackoverflow.com/questions/49259305/raise-valueerrornp-nan-is-an-invalid-document-expected-byte-or?rq=1
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna(' ')
```

In [43]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

#split the data into train and test fo bag of words

x_t,x_test,y_t,y_test=model_selection.train_test_split(project_data,y,test_size=0.3,random_state=0)
#split train into cross val train and cross val test
x_train,x_cv,y_train,y_cv=model_selection.train_test_split(x_t,y_t,test_size=0.3,random_state=0)
```

spliting train_data into train and cross validation in ratio of 7/3

In [44]:

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

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 encoding categorical features

In [45]:

```
x_train.head(2)
```

Out[45]:

	Unnamed: 0	id	teacher_id	teacher_prefix	sch
266	105761	p153429	21906b0de0445202f0a9823ee3aca7bf	Ms.	TN
106324	128977	p229920	a4782eb46f3f8b3bbd6853c1eefe2e00	Teacher	CO

In [46]:

```
#one hot encoding for clean_categories
#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer1 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
#vectorizer.fit(X_train['clean_subcategories'].values)
x_train_categories_one_hot = vectorizer1.fit_transform(x_train['clean_categories'].valu
es)
x_cv_categories_one_hot = vectorizer1.fit_transform(x_cv['clean_categories'].values)
x_test_categories_one_hot = vectorizer1.fit_transform(x_test['clean_categories'].values
)
print(vectorizer1.get_feature_names())
print("Shape of matrix after one hot encodig ",x_train_categories_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_cv_categories_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_test_categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (53531, 9)
Shape of matrix after one hot encodig  (22942, 9)
Shape of matrix after one hot encodig  (32775, 9)
```

In [47]:

```
#one hot encoding for clean_subcategories
#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer2 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
x_train_subcategories_one_hot = vectorizer2.fit_transform(x_train['clean_subcategories'
].values)
x_cv_subcategories_one_hot = vectorizer2.fit_transform(x_cv['clean_subcategories'].valu
es)
x_test_subcategories_one_hot = vectorizer2.fit_transform(x_test['clean_subcategories'].
values)
print(vectorizer2.get_feature_names())
print("Shape of matrix after one hot encodig ",x_train_subcategories_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_cv_subcategories_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_test_subcategories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (53531, 9)
Shape of matrix after one hot encodig  (22942, 9)
Shape of matrix after one hot encodig  (32775, 9)
```

In [48]:

```
#one hot encoding for school_state
#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer3 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
x_train_school_state_one_hot = vectorizer3.fit_transform(x_train['school_state'].values
)
x_cv_school_state_one_hot = vectorizer3.fit_transform(x_cv['school_state'].values)
x_test_school_state_one_hot = vectorizer3.fit_transform(x_test['school_state'].values)
print(vectorizer3.get_feature_names())
print("Shape of matrix after one hot encodig ",x_train_school_state_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_cv_school_state_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_test_school_state_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (53531, 9)
Shape of matrix after one hot encodig  (22942, 9)
Shape of matrix after one hot encodig  (32775, 9)
```

In [49]:

```
#one hot encoding for project_grade_category
#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer4 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
x_train_grade_category_one_hot = vectorizer4.fit_transform(x_train['project_grade_categ
ory'].values)
x_cv_grade_category_one_hot = vectorizer4.fit_transform(x_cv['project_grade_category'].
values)
x_test_grade_category_one_hot = vectorizer4.fit_transform(x_test['project_grade_categor
y'].values)
print(vectorizer4.get_feature_names())
print("Shape of matrix after one hot encodig ",x_train_grade_category_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_cv_grade_category_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_test_grade_category_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (53531, 9)
Shape of matrix after one hot encodig  (22942, 9)
Shape of matrix after one hot encodig  (32775, 9)
```

In [50]:

```
#one hot encoding for project_grade_category
#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer5 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)
x_train_prefix_one_hot = vectorizer5.fit_transform(x_train['teacher_prefix'].values)
x_cv_prefix_one_hot = vectorizer5.fit_transform(x_cv['teacher_prefix'].values)
x_test_prefix_one_hot = vectorizer5.fit_transform(x_test['teacher_prefix'].values)
print(vectorizer5.get_feature_names())
print("Shape of matrix after one hot encodig ",x_train_prefix_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_cv_prefix_one_hot.shape)
print("Shape of matrix after one hot encodig ",x_test_prefix_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (53531, 9)
Shape of matrix after one hot encodig  (22942, 9)
Shape of matrix after one hot encodig  (32775, 9)
```

In [51]:

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

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

2.2.2 encoding numerical features

In [52]:

```
x_train.head(2)
```

Out[52]:

	Unnamed: 0	id	teacher_id	teacher_prefix	sch
266	105761	p153429	21906b0de0445202f0a9823ee3aca7bf	Ms.	TN
106324	128977	p229920	a4782eb46f3f8b3bbd6853c1eefe2e00	Teacher	CO

In [53]:

```
'''#price standardization of x_train data
#-------------------------------------
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
x_train_price_standardized = price_scalar.transform(x_train['price'].values.reshape(-1, 1))'''
```

Out[53]:

```
'#price standardization of x_train data\n#-------------------------------
------\n# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=53
0s\n# standardization sklearn: https://scikit-learn.org/stable/modules/gen
erated/sklearn.preprocessing.StandardScaler.html\nfrom sklearn.preprocessi
ng import StandardScaler\n\n# price_standardized = standardScalar.fit(proj
ect_data[\'price\'].values)\n# this will rise the error\n# ValueError: Exp
ected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.
287.73   5.5 ].\n# Reshape your data either using array.reshape(-1, 1)\n\n
price_scalar = StandardScaler()\nprice_scalar.fit(x_train[\'price\'].value
s.reshape(-1,1)) # finding the mean and standard deviation of this data\np
rint(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price
_scalar.var_[0])}")\n\n# Now standardize the data with above maen and vari
ance.\nx_train_price_standardized = price_scalar.transform(x_train[\'price
\'].values.reshape(-1, 1))'
```

In [54]:

```
'''x_train_price_standardized = (x_train['price']-min(x_train['price']))/(max(x_train['price'])-min(x_train['price']))
x_train_price_standardized=x_train_price_standardized.values.reshape(24500,1)
print(type(x_train_price_standardized))
print(x_train_price_standardized.shape)'''
```

Out[54]:

```
"x_train_price_standardized = (x_train['price']-min(x_train['price']))/(ma
x(x_train['price'])-min(x_train['price']))\nx_train_price_standardized=x_t
rain_price_standardized.values.reshape(24500,1)\nprint(type(x_train_price_
standardized))\nprint(x_train_price_standardized.shape)"
```

In [55]:

```
#Normalize thae dataset
#https://www.w3cschool.cn/doc_scikit_learn/scikit_learn-modules-generated-sklearn-preprocessing-normalize.html
#https://stackoverflow.com/questions/53723928/attributeerror-series-object-has-no-attribute-reshape
import sklearn
x_train_price_standardized=sklearn.preprocessing.normalize(x_train['price'].values.reshape(-1,1), norm='l2', axis=1, copy=True, return_norm=False)
x_train_price_standardized.shape
```

Out[55]:

```
(53531, 1)
```

In [56]:

```
'''#price standardization of x_cv data
#-------------------------------------
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
x_cv_price_standardized = price_scalar.transform(x_cv['price'].values.reshape(-1,
 1))'''
```

Out[56]:

```
'#price standardization of x_cv data\n#-----------------------------------
---\n# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
\n# standardization sklearn: https://scikit-learn.org/stable/modules/gener
ated/sklearn.preprocessing.StandardScaler.html\nfrom sklearn.preprocessing
import StandardScaler\n\n# price_standardized = standardScalar.fit(project
_data[\'price\'].values)\n# this will rise the error\n# ValueError: Expect
ed 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.
287.73   5.5 ].\n# Reshape your data either using array.reshape(-1, 1)\n\n
price_scalar = StandardScaler()\nprice_scalar.fit(x_cv[\'price\'].values.r
eshape(-1,1)) # finding the mean and standard deviation of this data\n#pri
nt(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_s
calar.var_[0])}")\n\n# Now standardize the data with above maen and varian
ce.\nx_cv_price_standardized = price_scalar.transform(x_cv[\'price\'].valu
es.reshape(-1, 1))'
```

In [57]:

```
'''x_cv_price_standardized = (x_cv['price']-min(x_cv['price']))/(max(x_cv['price'])-min
(x_cv['price']))
x_cv_price_standardized=x_cv_price_standardized.values.reshape(10500,1)
print(type(x_cv_price_standardized))
print(x_cv_price_standardized.shape)'''
```

Out[57]:

```
"x_cv_price_standardized = (x_cv['price']-min(x_cv['price']))/(max(x_cv['p
rice'])-min(x_cv['price']))\nx_cv_price_standardized=x_cv_price_standardiz
ed.values.reshape(10500,1)\nprint(type(x_cv_price_standardized))\nprint(x_
cv_price_standardized.shape)"
```

In [58]:

```
#Normalize thae dataset
#https://www.w3cschool.cn/doc_scikit_learn/scikit_learn-modules-generated-sklearn-prepr
ocessing-normalize.html
#https://stackoverflow.com/questions/53723928/attributeerror-series-object-has-no-attri
bute-reshape
import sklearn
x_cv_price_standardized=sklearn.preprocessing.normalize(x_cv['price'].values.reshape(-1
,1), norm='l2', axis=1, copy=True, return_norm=False)
x_cv_price_standardized.shape
```

Out[58]:

```
(22942, 1)
```

In [59]:

```
'''#price standardization of x_test data
#-------------------------------------
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

# Now standardize the data with above maen and variance.
x_test_price_standardized = price_scalar.transform(x_test['price'].values.reshape(-1,
 1))'''
```

Out[59]:

```
'#price standardization of x_test data\n#-------------------------------
-----\n# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530
s\n# standardization sklearn: https://scikit-learn.org/stable/modules/gene
rated/sklearn.preprocessing.StandardScaler.html\nfrom sklearn.preprocessin
g import StandardScaler\n\n# price_standardized = standardScalar.fit(proje
ct_data[\'price\'].values)\n# this will rise the error\n# ValueError: Expe
cted 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.
287.73   5.5 ].\n# Reshape your data either using array.reshape(-1, 1)\n\n
price_scalar = StandardScaler()\nprice_scalar.fit(x_test[\'price\'].value
s.reshape(-1,1)) # finding the mean and standard deviation of this data\n#
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(pric
e_scalar.var_[0])}")\n\n# Now standardize the data with above maen and var
iance.\nx_test_price_standardized = price_scalar.transform(x_test[\'price
\'].values.reshape(-1, 1))'
```

In [60]:

```
'''x_test_price_standardized = (x_test['price']-min(x_test['price']))/(max(x_test['pric
e'])-min(x_test['price']))
x_test_price_standardized=x_test_price_standardized.values.reshape(15000,1)
print(type(x_test_price_standardized))
print(x_test_price_standardized.shape)'''
```

Out[60]:

```
"x_test_price_standardized = (x_test['price']-min(x_test['price']))/(max(x
_test['price'])-min(x_test['price']))\nx_test_price_standardized=x_test_pr
ice_standardized.values.reshape(15000,1)\nprint(type(x_test_price_standard
ized))\nprint(x_test_price_standardized.shape)"
```

In [61]:

```
'''x_test_price_standardized = (x_test['price']-min(x_test['price']))/(max(x_test['pric
e'])-min(x_test['price']))
x_test_price_standardized.reshape(-1,1).shape
print(type(x_test_price_standardized))
print(x_test_price_standardized.shape)'''
```

Out[61]:

```
"x_test_price_standardized = (x_test['price']-min(x_test['price']))/(max(x
_test['price'])-min(x_test['price']))\nx_test_price_standardized.reshape(-
1,1).shape\nprint(type(x_test_price_standardized))\nprint(x_test_price_sta
ndardized.shape)"
```

In [62]:

```
#Normalize thae dataset
#https://www.w3cschool.cn/doc_scikit_learn/scikit_learn-modules-generated-sklearn-prepr
ocessing-normalize.html
#https://stackoverflow.com/questions/53723928/attributeerror-series-object-has-no-attri
bute-reshape
import sklearn
x_test_price_standardized=sklearn.preprocessing.normalize(x_test['price'].values.reshap
e(-1,1), norm='l2', axis=1, copy=True, return_norm=False)
x_test_price_standardized.shape
```

Out[62]:

```
(32775, 1)
```

2.2.3 merge numerical and categorical data

In [63]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
x_train_ohe = hstack((x_train_categories_one_hot, x_train_subcategories_one_hot, x_trai
n_school_state_one_hot, x_train_grade_category_one_hot, x_train_prefix_one_hot, x_train
_price_standardized))
x_cv_ohe = hstack((x_cv_categories_one_hot, x_cv_subcategories_one_hot, x_cv_school_sta
te_one_hot, x_cv_grade_category_one_hot, x_cv_prefix_one_hot, x_cv_price_standardized))
x_test_ohe = hstack((x_test_categories_one_hot, x_test_subcategories_one_hot, x_test_sc
hool_state_one_hot, x_test_grade_category_one_hot, x_test_prefix_one_hot, x_test_price_
standardized))

print(x_train_ohe.shape)
print(x_cv_ohe.shape)
print(x_test_ohe.shape)
```

```
(53531, 46)
(22942, 46)
(32775, 46)
```

In [64]:

```
print(x_train_categories_one_hot.shape)
print(x_train_subcategories_one_hot.shape)
print(x_train_school_state_one_hot.shape)
print(x_train_grade_category_one_hot.shape)
print(x_train_prefix_one_hot.shape)
print(x_train_price_standardized.shape)
```

```
(53531, 9)
(53531, 9)
(53531, 9)
(53531, 9)
(53531, 9)
(53531, 1)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

2.4 Appling 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

In [65]:

```
# please write all the code with proper documentation, and proper titles for each subse
ction
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your
code

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

2.4.1 Applying Naive Bayes on BOW, SET 1

vectorize the essay and title data, SET 1

In [66]:

```
#you can vectorize the essay
#_____________________________
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_essay = CountVectorizer(min_df=10)
vectorizer_essay.fit(x_train['preprocessed_essays'].values)# fit has to apply only on train data
z_bow1=vectorizer_essay.fit(x_train['preprocessed_essays'].values)# fit has to apply only on train data

# we use fitted CountVectorizer to convert the text to vector
x_train_bow_essays = vectorizer_essay.transform(x_train['preprocessed_essays'].values)
x_cv_bow_essays = vectorizer_essay.transform(x_cv['preprocessed_essays'].values)
x_test_bow_essays = vectorizer_essay.transform(x_test['preprocessed_essays'].values)

print("Shape of matrix after one hot encodig ",x_train_bow_essays.shape, y_train.shape)
print("Shape of matrix after one hot encodig ",x_cv_bow_essays.shape)
print("Shape of matrix after one hot encodig ",x_test_bow_essays.shape)
```

```
Shape of matrix after one hot encodig  (53531, 12411) (53531,)
Shape of matrix after one hot encodig  (22942, 12411)
Shape of matrix after one hot encodig  (32775, 12411)
```

In [67]:

```
#https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
#you can vectorize the title
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_title = CountVectorizer(min_df=10)
vectorizer_title.fit(x_train['preprocessed_project_title'].values)# fit has to apply only on train data
z_bow2=vectorizer_title.fit(x_train['preprocessed_project_title'].values)# fit has to apply only on train data

# we use fitted CountVectorizer to convert the text to vector
x_train_bow_title = vectorizer_title.transform(x_train['preprocessed_project_title'].values)
x_cv_bow_title = vectorizer_title.transform(x_cv['preprocessed_project_title'].values)
x_test_bow_title = vectorizer_title.transform(x_test['preprocessed_project_title'].values)

print("Shape of matrix after one hot encodig ",x_train_bow_title.shape)
print("Shape of matrix after one hot encodig ",x_cv_bow_title.shape)
print("Shape of matrix after one hot encodig ",x_test_bow_title.shape)
```

```
Shape of matrix after one hot encodig  (53531, 2193)
Shape of matrix after one hot encodig  (22942, 2193)
Shape of matrix after one hot encodig  (32775, 2193)
```

In [68]:

```
x_train.columns
```

Out[68]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_essay_1', 'project_essay
_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'preprocessed_essays', 'preprocessed_project_title'],
      dtype='object')
```

In [69]:

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

merge dataset, SET 1

In [70]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
x_train_bow = hstack((x_train_ohe, x_train_bow_essays, x_train_bow_title)).tocsr()
x_cv_bow = hstack((x_cv_ohe, x_cv_bow_essays, x_cv_bow_title)).tocsr()
x_test_bow = hstack((x_test_ohe, x_test_bow_essays, x_test_bow_title)).tocsr()


print(x_train_bow.shape)
print(x_cv_bow.shape)
print(x_test_bow.shape)
```

```
(53531, 14650)
(22942, 14650)
(32775, 14650)
```

In [71]:

```
type(x_train_bow)
```

Out[71]:

```
scipy.sparse.csr.csr_matrix
```

simple tuning

In [72]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 =
 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

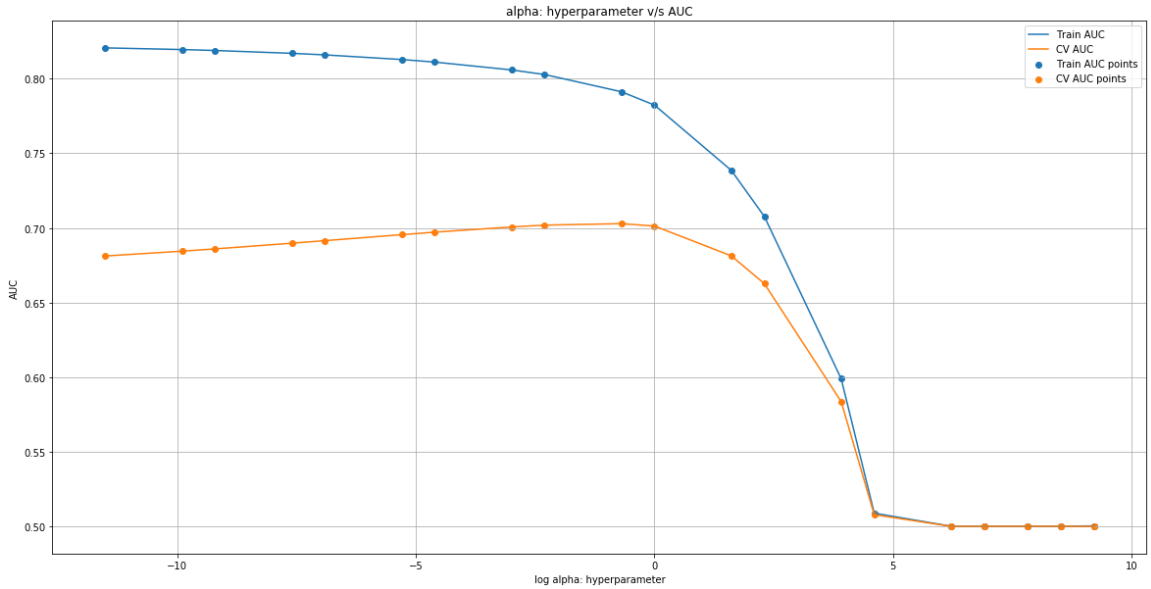
In [73]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_alphas = []

alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 1
0, 50, 100, 500, 1000, 2500, 5000, 10000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i)
    nb.fit(x_train_bow, y_train)

    y_train_pred = batch_predict(nb, x_train_bow)
    y_cv_pred = batch_predict(nb, x_cv_bow)


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

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)
```

```
100%|████████████████████████████████████████████████████████████
████████████| 20/20 [00:02<00:00,  8.48it/s]
100%|████████████████████████████████████████████████████████████
████████████████████| 20/20 [00:00<?, ?it/s]
```

In [74]:

```
plt.figure(figsize=(20,10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Observation:

both maximum and minimum value of alpha not good for model, for more value of alpha AUC is very low and for low value of alpha Overfitting occure. so alpha will be in between near to 0.5

Grid search, SET 1

In [75]:

```
#https://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/
# Grid Search for Algorithm Tuning
import numpy as np
from sklearn import datasets
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

# prepare a range of alpha values to test
#alphas = np.array([1,0.1,0.01,0.001,0.0001])
# create and fit a ridge regression model, testing each alpha
parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1,
0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]}

n_folds = 10
my_cv = TimeSeriesSplit(n_splits=n_folds).split(x_train_bow)

model = MultinomialNB()
grid = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas),cv=my_cv, scoring='r
oc_auc')
grid.fit(x_train_bow, y_train)
print(grid)
# summarize the results of the grid search
print(grid.best_score_)
print(grid.best_estimator_.alpha)

#results_grid_bow_NB = pd.DataFrame.from_dict(grid.cv_results_).sort_values(['alpha'])

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

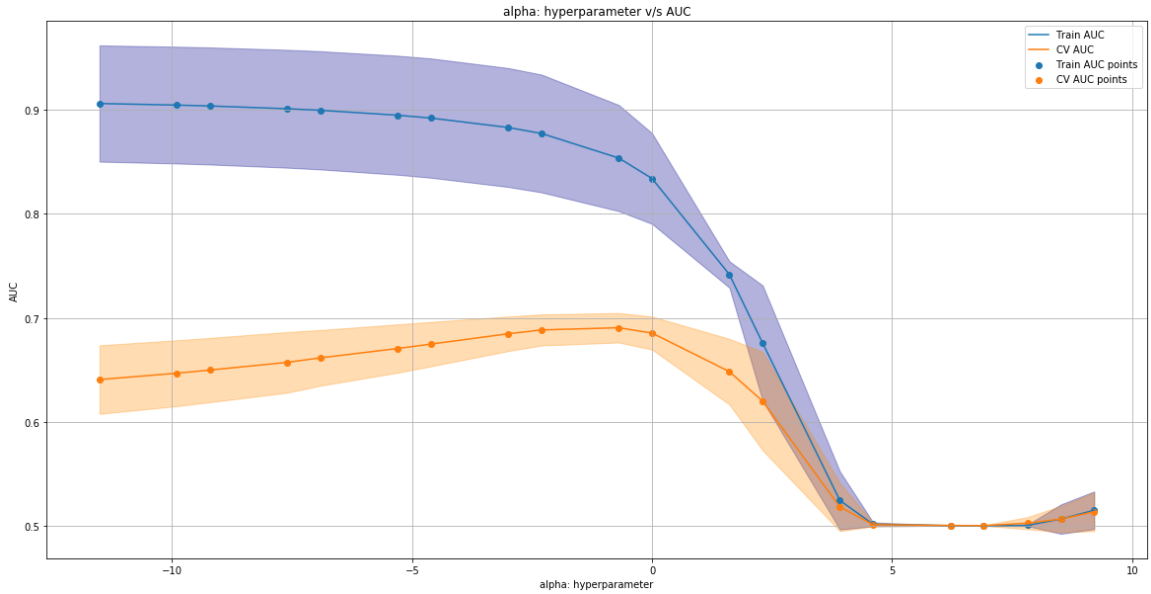
```
GridSearchCV(cv=<generator object TimeSeriesSplit.split at 0x000001D88903C
CA8>,
       error_score='raise',
       estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'alpha': [1e-05, 5e-05, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=0)
0.6904429102737408
0.5
```

optimal alpha value is 0.0001 nearly zero, where F1 score of 0.85(1) on BOW train dataset.

In [76]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 1
0, 50, 100, 500, 1000, 2500, 5000, 10000]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(20,10))

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

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

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


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```

```
100%|██████████| 20/20 [00:00<?, ?it/s]
```



In [77]:

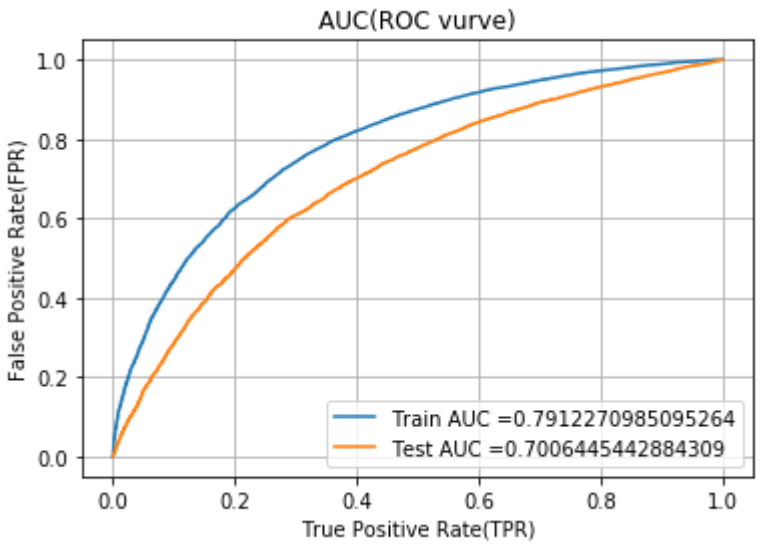
```
alpha_opt_bow=0.5
```

Apply best hyperparameter on test dataset, SET 1

You need to plot the performance of model both on train data and cross validation data for each hyper parameter, as shown in the figure
Once you find the best hyper parameter, you need to train your model-M using the best hyper-param. Now, find the AUC on test data and plot the ROC curve on both train and test using model-M.
Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points

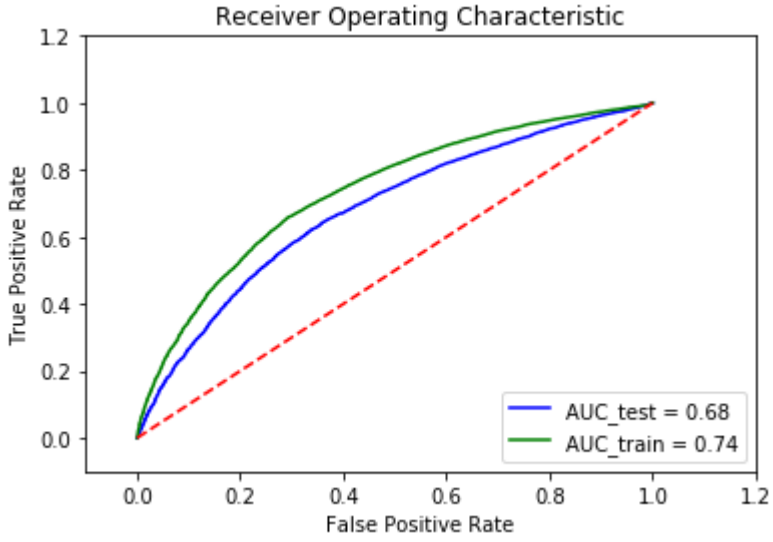
In [78]:

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

nb_bow = MultinomialNB(alpha = alpha_opt_bow)

nb_bow.fit(x_train_bow, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(nb_bow, x_train_bow)
y_test_pred = batch_predict(nb_bow, x_test_bow)

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)

BOW_roc_auc_train = auc(test_fpr, test_tpr)
BOW_roc_auc_test = auc(train_fpr, train_tpr)

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



In [79]:

```
'''# Plotting the ROC Curve for the Best Classifier
#_____________________________________________________
#https://datamize.wordpress.com/2015/01/24/how-to-plot-a-roc-curve-in-scikit-learn/
from sklearn.metrics import roc_curve, auc
Y_score_test = grid.best_estimator_.predict_proba(x_test_bow)
fpr1, tpr1, thresholds1 = roc_curve(y_test,Y_score_test[:, 1])
BOW_roc_auc_test = auc(fpr1, tpr1)

Y_score_train = grid.best_estimator_.predict_proba(x_train_bow)
fpr2, tpr2, thresholds2 = roc_curve(y_train,Y_score_train[:, 1])
BOW_roc_auc_train = auc(fpr2, tpr2)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr1, tpr1, 'b',label='AUC_test = %0.2f'% BOW_roc_auc_test)
plt.plot(fpr2, tpr2, 'g',label='AUC_train = %0.2f'% BOW_roc_auc_train)
plt.legend(loc='lower right')
plt.plot([0,1],[0,1],'r--')
plt.xlim([-0.1,1.2])
plt.ylim([-0.1,1.2])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()'''
```



Here AUC value on BOW test dataset is 0.68. Model is better then BOW because AUC of both train and test are near hence, they are neither underfit or overfit.

In [80]:

```
'''# Display Performance of the  Hyper-parametrized BOW model on TEST data


y_pred = grid.best_estimator_.predict(x_test_bow)

#Evaluate the model accuracy on TEST data

test_accuracy_bow = accuracy_score(y_test, y_pred, normalize=True) * 100
points = accuracy_score(y_test, y_pred, normalize=False)

# Display the classification report
print(classification_report(y_test, y_pred,digits=4))

#Display the model accuracy on TEST data
print('\nThe number of accurate predictions out of {} data points on TEST data is {}'.f
ormat(x_test_bow.shape[0], points))
print('Accuracy of the {} model on TEST data is {} %'.format("BOW", '{:f}'.format(np.ro
und(test_accuracy_bow,2))))'''
```

```
              precision    recall  f1-score   support

           0     0.2612    0.5283    0.3495      4993
           1     0.8961    0.7314    0.8054     27782

   micro avg     0.7004    0.7004    0.7004     32775
   macro avg     0.5786    0.6299    0.5775     32775
weighted avg     0.7994    0.7004    0.7360     32775


The number of accurate predictions out of 32775 data points on TEST data i
s 22957
Accuracy of the BOW model on TEST data is 70.040000 %
```

confusion matrix(test)

In [79]:

```
def predict(proba, threshould, fpr, tpr):

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

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

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
nd(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

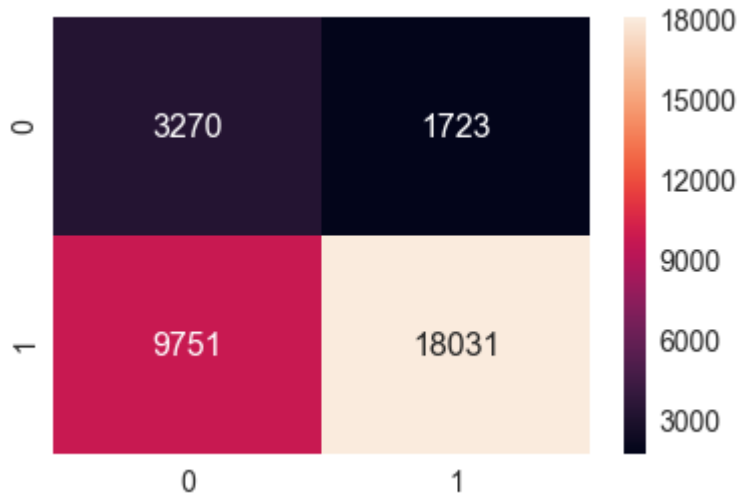
In [80]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_tpr
)))


conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, te_thr
esholds, test_fpr, test_tpr)), range(2),range(2))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================
==========================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4277518148905696 for threshold 0.929
[[ 3270  1723]
 [ 9751 18031]]
the maximum value of tpr*(1-fpr) 0.4277518148905696 for threshold 0.929
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d89929bda0>
```



2.4.1.1 Top 10 important features of positive class from SET 1

In [81]:

```
'''#https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive
-bayes#50530697
#Note : Putting a - sign indicates the indexes will be sorted in descending order.
pos_class_prob_sorted = (-grid.best_estimator_.feature_log_prob_[1, :]).argsort()
pos_class_top10_features = np.take(z_bow1.get_feature_names(), pos_class_prob_sorted[:1
0])
print("The top 10 most frequent words from the positive class are :\n")
print(pos_class_top10_features)'''
```

Out[81]:

```
'#https://stackoverflow.com/questions/50526898/how-to-get-feature-importan
ce-in-naive-bayes#50530697\n#Note : Putting a - sign indicates the indexes
will be sorted in descending order.\npos_class_prob_sorted = (-grid.best_e
stimator_.feature_log_prob_[1, :]).argsort()\npos_class_top10_features = n
p.take(z_bow1.get_feature_names(), pos_class_prob_sorted[:10])\nprint("The
top 10 most frequent words from the positive class are :\n")\nprint(pos_cl
ass_top10_features)'
```

In [95]:

```
print(x_train_bow.shape, y_train.shape)
print(x_cv_bow.shape)
print(x_test_bow.shape)
```

```
(53531, 14650) (53531,)
(22942, 14650)
(32775, 14650)
```

In [124]:

```
#probability value of positive class
nb_bow = MultinomialNB(alpha = alpha_opt_bow)
nb_bow.fit(x_train_bow, y_train)

bow_features_probs_neg = {}
for a in range(x_train_bow.shape[1]) :
    bow_features_probs_neg[a] = nb_bow.feature_log_prob_[0,a]

len(bow_features_probs_neg.values())
```

Out[124]:

```
14650
```

In [97]:

```
#adding categorical variable name
bow_features_names=[]
for a in vectorizer1.get_feature_names() :
    bow_features_names.append(a)
for a in vectorizer2.get_feature_names() :
    bow_features_names.append(a)
for a in vectorizer3.get_feature_names() :
    bow_features_names.append(a)
for a in vectorizer4.get_feature_names() :
    bow_features_names.append(a)
for a in vectorizer5.get_feature_names() :
    bow_features_names.append(a)

len(bow_features_names)
```

Out[97]:

```
45
```

In [98]:

```
# adding numerical
bow_features_names.append("price")
'''bow_features_names.append("quantity")
bow_features_names.append("price")
bow_features_names.append("price")
bow_features_names.append("price")'''
```

Out[98]:

```
'bow_features_names.append("quantity")\nbow_features_names.append("price")
\nbow_features_names.append("price")\nbow_features_names.append("price")'
```

In [99]:

```
for a in z_bow1.get_feature_names() :
    bow_features_names.append(a)

for a in z_bow2.get_feature_names() :
    bow_features_names.append(a)
```

In [100]:

```
len(bow_features_names)
```

Out[100]:

```
14650
```

In [102]:

```
final_bow_features = pd.DataFrame({'feature_prob_estimates' : list(bow_features_probs_n
eg.values()), 'feature_names' : bow_features_names})
```

In [147]:

```
#final_bow_features
```

In [131]:

```
neg = final_bow_features.sort_values(by = ['feature_prob_estimates'], ascending = True)
```

2.4.1.2 Top 10 important features of negative class from SET 1

In [132]:

```
print('Top 10 negative feature' )
neg.head(10)
```

```
Top 10 negative feature
```

Out[132]:

	feature_prob_estimates	feature_names
12690	-14.603237	brand
5652	-14.603237	implements
10956	-14.603237	sweatshirts
2789	-14.603237	crossfit
2788	-14.603237	crosses
844	-14.603237	aristotle
9099	-14.603237	redirected
10466	-14.603237	splitting
848	-14.603237	arledge
7881	-14.603237	paddle

top 10 important feature of negative class vectorised from essay dataset

2.4.1.2 Top 10 important features of positive class from SET 1

In [133]:

```
bow_features_probs_positive = {}

for a in range(x_train_bow.shape[1]):
    bow_features_probs_positive[a] = nb_bow.feature_log_prob_[1,a]
```

In [134]:

```
final_bow_features_positive = pd.DataFrame({'feature_prob_estimates' : list(bow_feature
s_probs_positive.values()), 'feature_names' : bow_features_names})
```

In [135]:

```
pos = final_bow_features_positive.sort_values(by = ['feature_prob_estimates'], ascendin
g = True)
```

In [136]:

```
pos.head(10)
```

Out[136]:

	feature_prob_estimates	feature_names
28	-16.386061	Care_Hunger
20	-16.386061	History_Civics
21	-16.386061	Music_Arts
22	-16.386061	AppliedLearning
23	-16.386061	SpecialNeeds
24	-16.386061	Health_Sports
25	-16.386061	Math_Science
26	-16.386061	Literacy_Language
27	-16.386061	Warmth
37	-16.386061	Care_Hunger

2.4.2 Applying Naive Bayes on TFIDF, SET 2

TFIDF Vectorizing essy and title variable, SET 2

In [0]:

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

In [83]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf = TfidfVectorizer(min_df=10,  ngram_range=(1,4), max_features=5000)
vectorizer_tfidf.fit(x_train['preprocessed_essays'].values)# fit has to apply only on t
rain data
z_tfidf1=vectorizer_tfidf.fit(x_train['preprocessed_essays'].values)# fit has to apply
 only on train data

# we use fitted CountVectorizer to convert the text to vector
x_train_tfidf_essays = vectorizer_tfidf.transform(x_train['preprocessed_essays'].values
)
x_cv_tfidf_essays = vectorizer_tfidf.transform(x_cv['preprocessed_essays'].values)
x_test_tfidf_essays = vectorizer_tfidf.transform(x_test['preprocessed_essays'].values)

print("Shape of matrix after one hot encodig ",x_train_tfidf_essays.shape, y_train.shap
e)
print("Shape of matrix after one hot encodig ",x_cv_tfidf_essays.shape)
print("Shape of matrix after one hot encodig ",x_test_tfidf_essays.shape)
```

```
Shape of matrix after one hot encodig  (53531, 5000) (53531,)
Shape of matrix after one hot encodig  (22942, 5000)
Shape of matrix after one hot encodig  (32775, 5000)
```

In [84]:

```
#TFIDF Vectorizer on `project_title`

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf = TfidfVectorizer(min_df=10, ngram_range=(1,4), max_features=5000)
vectorizer_tfidf.fit(x_train['preprocessed_project_title'].values)# fit has to apply on
ly on train data
z_tfidf2=vectorizer_tfidf.fit(x_train['preprocessed_project_title'].values)# fit has to
apply only on train data

# we use fitted CountVectorizer to convert the text to vector
x_train_tfidf_title = vectorizer_tfidf.transform(x_train['preprocessed_project_title'].
values)
x_cv_tfidf_title = vectorizer_tfidf.transform(x_cv['preprocessed_project_title'].values
)
x_test_tfidf_title = vectorizer_tfidf.transform(x_test['preprocessed_project_title'].va
lues)

print("Shape of matrix after one hot encodig ",x_train_tfidf_title.shape)
print("Shape of matrix after one hot encodig ",x_cv_tfidf_title.shape)
print("Shape of matrix after one hot encodig ",x_test_tfidf_title.shape)
```

```
Shape of matrix after one hot encodig  (53531, 4452)
Shape of matrix after one hot encodig  (22942, 4452)
Shape of matrix after one hot encodig  (32775, 4452)
```

merge all sparse data, SET 2

In [85]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
x_train_tfidf = hstack((x_train_ohe, x_train_tfidf_essays, x_train_tfidf_title)).tocsr
()
x_cv_tfidf = hstack((x_cv_ohe, x_cv_tfidf_essays, x_cv_tfidf_title)).tocsr()
x_test_tfidf = hstack((x_test_ohe, x_test_tfidf_essays, x_test_tfidf_title)).tocsr()

print(x_train_tfidf.shape)
print(x_cv_tfidf.shape)
print(x_test_tfidf.shape)
```

```
(53531, 9498)
(22942, 9498)
(32775, 9498)
```

simple tuning

In [86]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 =
 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

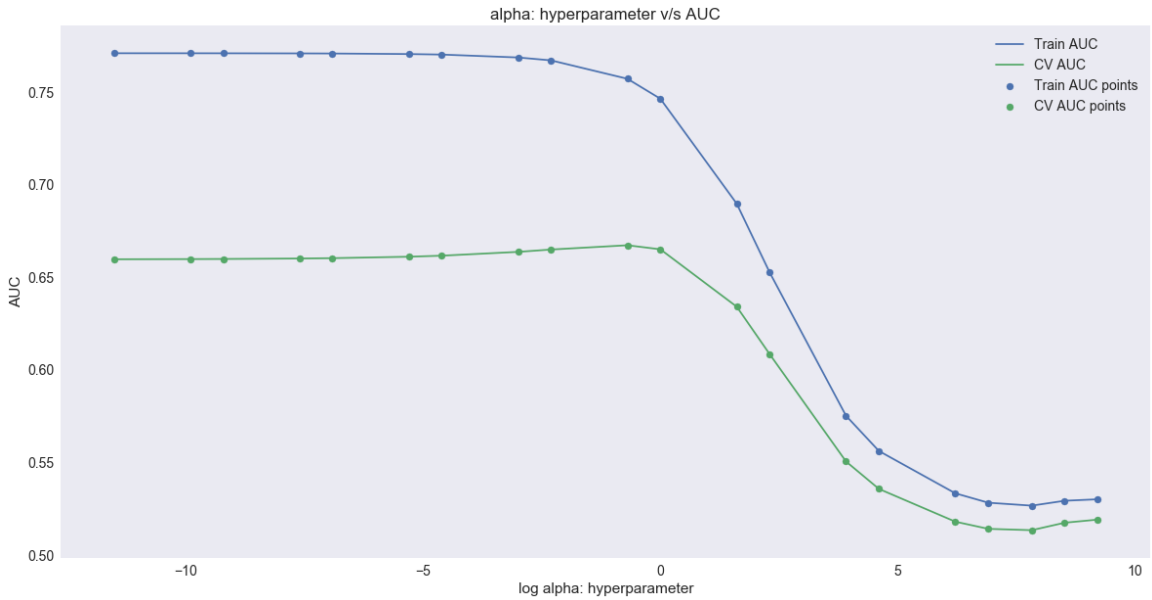
In [87]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_alphas = []

alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 1
0, 50, 100, 500, 1000, 2500, 5000, 10000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i)
    nb.fit(x_train_tfidf, y_train)

    y_train_pred = batch_predict(nb, x_train_tfidf)
    y_cv_pred = batch_predict(nb, x_cv_tfidf)


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

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)
```

```
100%|████████████████████████████████████████████████████████████
████████████| 20/20 [00:04<00:00,  5.09it/s]
100%|████████████████████████████████████████████████████████████
████████████████████| 20/20 [00:00<?, ?it/s]
```

In [88]:

```
plt.figure(figsize=(20,10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Observation: Same problem as in case of BOW, so here alpha will be 0.5.

Grid search, SET 2

In [89]:

```
#https://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/
# Grid Search for Algorithm Tuning
import numpy as np
from sklearn import datasets
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

# prepare a range of alpha values to test
#alphas = np.array([1,0.1,0.01,0.001,0.0001])
# create and fit a ridge regression model, testing each alpha
parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1,
0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]}

n_folds = 10
my_cv = TimeSeriesSplit(n_splits=n_folds).split(x_train_tfidf)

model = MultinomialNB()
grid = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas),cv=my_cv, scoring='r
oc_auc')
grid.fit(x_train_tfidf, y_train)
print(grid)
# summarize the results of the grid search
print(grid.best_score_)
print(grid.best_estimator_.alpha)

#results_grid_bow_NB = pd.DataFrame.from_dict(grid.cv_results_).sort_values(['alpha'])

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

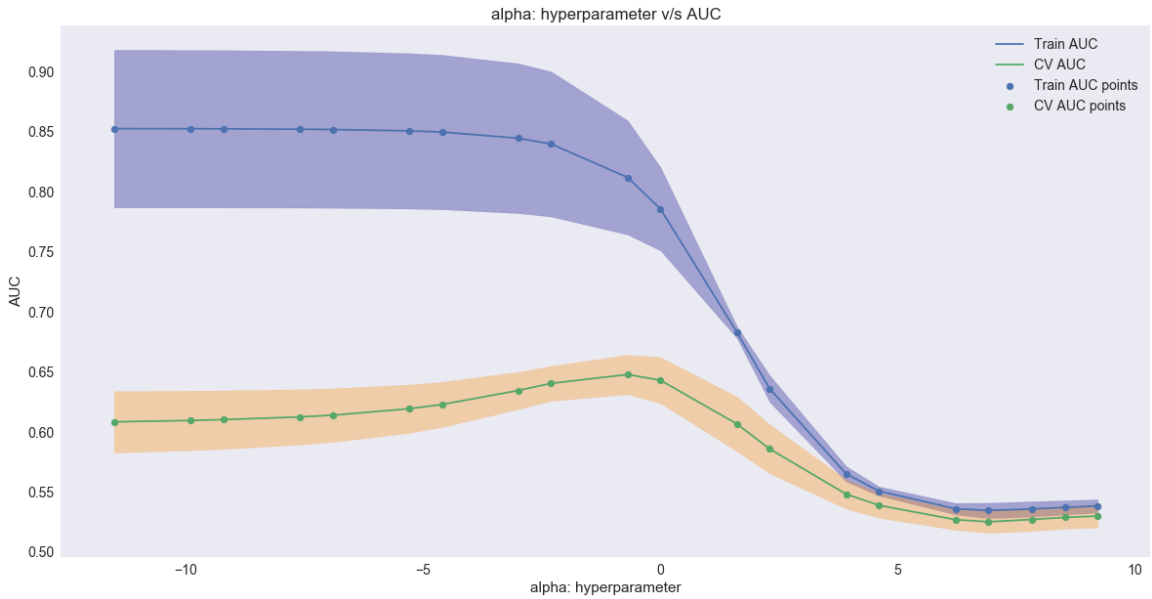
```
GridSearchCV(cv=<generator object TimeSeriesSplit.split at 0x000001D92759D
BA0>,
       error_score='raise',
       estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'alpha': [1e-05, 5e-05, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 2500, 5000, 10000]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=0)
0.6478557910449321
0.5
```

optimal value of alpha value in TFIDF train dataset is 1

In [90]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 1
0, 50, 100, 500, 1000, 2500, 5000, 10000]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(20,10))

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

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

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


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```

```
100%|██████████| 20/20 [00:00<00:00, 20078.05it/s]
```



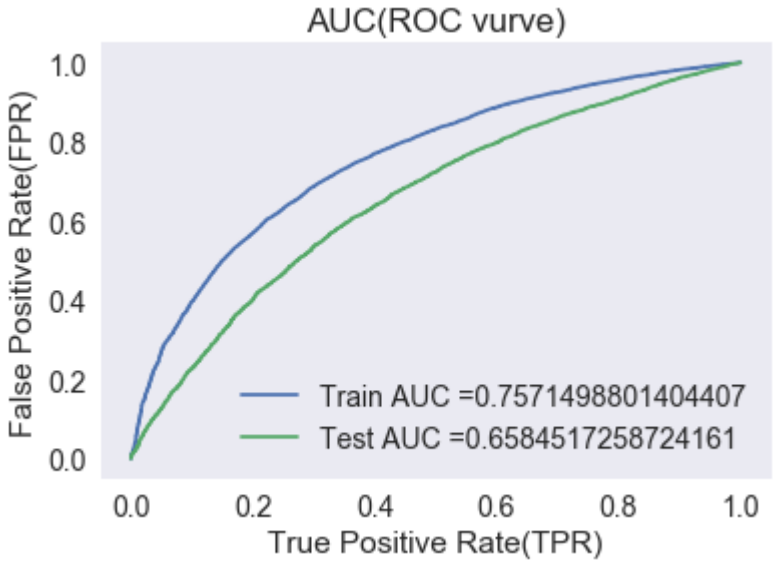
In [91]:

```
alpha_opt_tfidf=0.5
```

Apply best parameter on test data, SET 2

In [92]:

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

nb_tfidf = MultinomialNB(alpha = alpha_opt_tfidf)

nb_tfidf.fit(x_train_tfidf, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(nb_tfidf, x_train_tfidf)
y_test_pred = batch_predict(nb_tfidf, x_test_tfidf)

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)

tfidf_roc_auc_train = auc(test_fpr, test_tpr)
tfidf_roc_auc_test = auc(train_fpr, train_tpr)

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



Here AUC value on TFIDF test dataset is 0.67. Model is better then BOW because AUC of both train and test are near hence, they are neither underfit or overfit.

Confusing matrix(test)

In [93]:

```
def predict(proba, threshould, fpr, tpr):

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

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

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
nd(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

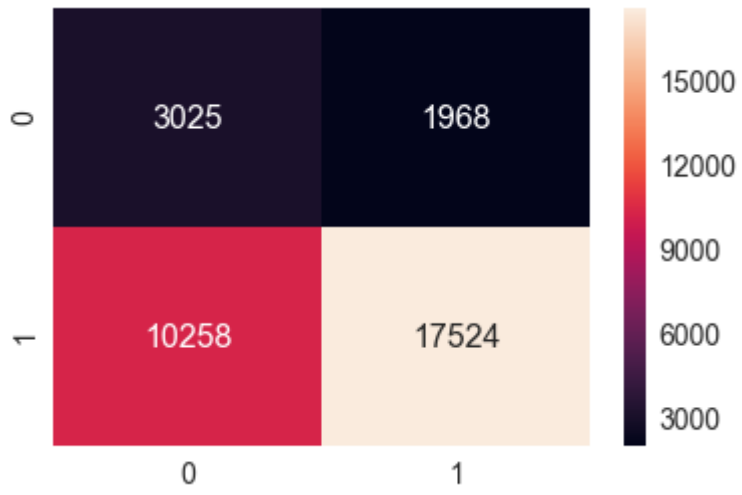
In [94]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, te_thresholds, test_fpr, test_tpr
)))


conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, te_thr
esholds, test_fpr, test_tpr)), range(2),range(2))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================
==========================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3869883678341817 for threshold 0.858
[[ 3025  1968]
 [10258 17524]]
the maximum value of tpr*(1-fpr) 0.3869883678341817 for threshold 0.858
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d9662cb8d0>
```



2.4.2.1 Top 10 important features of negative class from SET 2

In [141]:

```
print(x_train_tfidf.shape,y_train.shape)
print(x_cv_tfidf.shape)
print(x_test_tfidf.shape)
```

```
(53531, 9498) (53531,)
(22942, 9498)
(32775, 9498)
```

In [142]:

```
#https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-ba
yes#50530697
#Note : Putting a - sign#probability value of positive class
nb_tfidf = MultinomialNB(alpha = alpha_opt_tfidf)
nb_tfidf.fit(x_train_bow, y_train)

tfidf_features_probs_neg = {}
for a in range(x_train_tfidf.shape[1]) :
    tfidf_features_probs_neg[a] = nb_tfidf.feature_log_prob_[0,a]

len(tfidf_features_probs_neg)
```

Out[142]:

```
9498
```

In [143]:

```
#adding categorical variable name
tfidf_features_names=[]
for a in vectorizer1.get_feature_names() :
    tfidf_features_names.append(a)
for a in vectorizer2.get_feature_names() :
    tfidf_features_names.append(a)
for a in vectorizer3.get_feature_names() :
    tfidf_features_names.append(a)
for a in vectorizer4.get_feature_names() :
    tfidf_features_names.append(a)
for a in vectorizer5.get_feature_names() :
    tfidf_features_names.append(a)

len(tfidf_features_names)
```

Out[143]:

```
45
```

In [144]:

```
# adding numerical
tfidf_features_names.append("price")
```

In [145]:

```
for a in z_tfidf1.get_feature_names() :
    tfidf_features_names.append(a)

for a in z_tfidf2.get_feature_names() :
    tfidf_features_names.append(a)
```

In [146]:

```
len(tfidf_features_names)
```

Out[146]:

9498

In [147]:

```
#final tfidf feature
final_tfidf_features = pd.DataFrame({'feature_prob_estimates' : list(tfidf_features_pro
bs_neg.values()), 'feature_names' : tfidf_features_names})

neg = final_tfidf_features.sort_values(by = ['feature_prob_estimates'], ascending = Tru
e)
```

In [148]:

```
print('Top 10 negative feature' )
neg.head(10)
```

Top 10 negative feature

Out[148]:

	feature_prob_estimates	feature_names
892	-14.603237	community students
1037	-14.603237	culture
451	-14.603237	basic needs
4567	-14.603237	technology skills
3457	-14.603237	public school
2076	-14.603237	hours
3459	-14.603237	publish
3449	-14.603237	provide students creative meaningful
7033	-14.603237	kids are
4552	-14.603237	techniques

2.4.2.2 Top 10 important features of positive class from SET 2

In [149]:

```
tfidf_features_probs_positive = {}

for a in range(x_train_tfidf.shape[1]):
    tfidf_features_probs_positive[a] = nb_tfidf.feature_log_prob_[1,a]
```

In [150]:

```
final_tfidf_features_positive = pd.DataFrame({'feature_prob_estimates' : list(tfidf_fea
tures_probs_positive.values()), 'feature_names' : tfidf_features_names})
pos = final_tfidf_features_positive.sort_values(by = ['feature_prob_estimates'], ascend
ing = True)
```

In [151]:

```
pos.head(10)
```

Out[151]:

	feature_prob_estimates	feature_names
38	-16.386061	History_Civics
24	-16.386061	Health_Sports
25	-16.386061	Math_Science
26	-16.386061	Literacy_Language
27	-16.386061	Warmth
28	-16.386061	Care_Hunger
29	-16.386061	History_Civics
30	-16.386061	Music_Arts
31	-16.386061	AppliedLearning
32	-16.386061	SpecialNeeds

Conclusion

In [0]:

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

In [153]:

```
#!pip install prettytable
```

```
Collecting prettytable
  Downloading https://files.pythonhosted.org/packages/ef/30/4b0746848746ed
5941f052479e7c23d2b56d174b82f4fd34a25e389831f5/prettytable-0.7.2.tar.bz2
Building wheels for collected packages: prettytable
  Running setup.py bdist_wheel for prettytable: started
  Running setup.py bdist_wheel for prettytable: finished with status 'don
e'
  Stored in directory: C:\Users\Prof Arkopal Goswami\AppData\Local\pip\Cac
he\wheels\80\34\1c\3967380d9676d162cb59513bd9dc862d0584e045a162095606
Successfully built prettytable
Installing collected packages: prettytable
Successfully installed prettytable-0.7.2

wxpython 4.0.3 requires PyPubSub, which is not installed.
distributed 1.21.8 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pi
p' command.
```

In [157]:

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper parameter_alpha", "AUC_test","AUC_Train"
]
x.add_row(["BOW", "MultinomialNB", alpha_opt_bow, BOW_roc_auc_test,BOW_roc_auc_train])
x.add_row(["TFIDF", "MultinomialNB", alpha_opt_tfidf, tfidf_roc_auc_test,tfidf_roc_auc_
train])
print(x)

with open('Result_DonorsChoose_NB.txt', 'w') as w:
    w.write(str(x))
```

```
+------------+---------------+-----------------------+--------------------
+--------------------+
| Vectorizer |     Model     | Hyper parameter_alpha |      AUC_test
|     AUC_Train      |
+------------+---------------+-----------------------+--------------------
+--------------------+
|    BOW     | MultinomialNB |          0.5          | 0.7912270985095264
| 0.7006445442884309 |
|   TFIDF    | MultinomialNB |          0.5          | 0.7571498801404407
| 0.6584517258724161 |
+------------+---------------+-----------------------+--------------------
+--------------------+
```

Observation:

At alpha=0.5, both case is neither overfit nor underfit

Test accuracy in both case is nearly same.

Since AUC value in both case nearly same, where training AUC is less than test AUC, this model is not overfitting, Underfitting need further varification.