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 [77]:

```
%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 [78]:

```
#!pip install plotly
```

In [79]:

```
#!pip install gensim
```

In [80]:

```
#!pip install tqdm
```

1.1 Reading Data

In [81]:

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

In [82]:

```
project_data.shape
```

Out[82]:

```
(109248, 17)
```

In [83]:

```
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 [84]:

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

Out[84]:

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

In [85]:

```
# 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[85]:

	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 [86]:

```
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[86]:

	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 [87]:

```
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 [88]:

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

Out[88]:

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

In [89]:

```
#project_data = project_data.sample(n=50000)
#project_data=project_data.tail(1000)
project_data.shape
```

Out[89]:

```
(109248, 19)
```

In [90]:

```
project_data.columns
```

Out[90]:

```
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 [91]:

```
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 [92]:

```
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 [93]:

```
project_data.columns
```

Out[93]:

```
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 [94]:

```
# 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 [95]:

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

Out[95]:

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

In [96]:

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

In [97]:

```
'''# 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)'''
```

Out[97]:

```
'# printing some random reviews\nprint(project_data[\'essay\'].values[0])
\nprint("="*50)\nprint(project_data[\'essay\'].values[150])\nprint("="*50)
\nprint(project_data[\'essay\'].values[1000])\nprint("="*50)\nprint(projec
t_data[\'essay\'].values[20000])\nprint("="*50)\nprint(project_data[\'essa
y\'].values[99999])\nprint("="*50)'
```

In [98]:

```
# 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 [99]:

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

```
\"Creativity is intelligence having fun.\" --Albert Einstein. Our elementa
ry library at Greenville Elementary is anything but a quiet, hushed space.
It is a place for collaboration and research. It is a place for incorporat
ing technology. It is a place for innovation. And it is a place for creati
ng.Our school serves 350 third and fourth graders who primarily live in ru
ral and poverty-stricken areas in our community. Being a Title I school, a
pproximately 85% of them receive free or reduced lunch. But they are inqui
sitive, creative, and eager to learn. They love visiting the library to ch
eck out books, hear \r\nstories, create digital stories, and use the compu
ter lab for learning and fun. We want to build our library is Makerspace w
ith activities revolving around art and literacy to provide more engaging,
hands-on activities.We want to begin \"Makerspace Fridays!\" Our school re
cently received a $1000 grant for books for our arts-integrated Makerspac
e. We have received titles such as \"Origami for Everyone,\" \"How to Make
Stuff with Ducktape,\" and \"Cool Engineering Activities for Girls.\"  We
now need supplies to correlate with these new informational texts. By addi
ng these art and craft supplies, students will be able to design and creat
e masterpieces related to their coursework. \r\n\r\nFor example, while stu
dying Native Americans, students can use the looms and yarn to recreate Na
vajo and/or Pueblo weaving. Weaving can also be integrated with literacy t
hrough Greek mythology and the story of Arachne.\r\n\r\nCreating art with
perler beads has many possibilities! Students can design their own animals
after studying their characteristics. They can use symmetry and patterning
to create one-of-a-kind originals. \r\n\r\nOrigami reinforces geometry, th
inking skills, fractions, problem-solving, and just fun science!Our studen
ts need to be able to apply what they read and learn. If they read a how-t
o book, they will apply that reading through a hands-on art activity and a
ctually create a product. This is a crucial skill in the real world. By cr
eating and designing their own masterpieces, they are using many critical
thinking skills. Students will become more analytical thinkers.
==================================================
```

In [100]:

```
# \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)
```

```
 Creativity is intelligence having fun.  --Albert Einstein. Our elementary library at Greenville Elementary is anything but a quiet, hushed space. It is a place for collaboration and research. It is a place for incorporating technology. It is a place for innovation. And it is a place for creating.Our school serves 350 third and fourth graders who primarily live in rural and poverty-stricken areas in our community. Being a Title I school, approximately 85% of them receive free or reduced lunch. But they are inquisitive, creative, and eager to learn. They love visiting the library to check out books, hear   stories, create digital stories, and use the computer lab for learning and fun. We want to build our library is Makerspace with activities revolving around art and literacy to provide more engaging, hands-on activities.We want to begin  Makerspace Fridays!  Our school recently received a $1000 grant for books for our arts-integrated Makerspace. We have received titles such as  Origami for Everyone,   How to Make Stuff with Ducktape,  and  Cool Engineering Activities for Girls.   We now need supplies to correlate with these new informational texts. By adding these art and craft supplies, students will be able to design and create masterpieces related to their coursework.     For example, while studying Native Americans, students can use the looms and yarn to recreate Navajo and/or Pueblo weaving. Weaving can also be integrated with literacy through Greek mythology and the story of Arachne.    Creating art with perler beads has many possibilities! Students can design their own animals after studying their characteristics. They can use symmetry and patterning to create one-of-a-kind originals.     Origami reinforces geometry, thinking skills, fractions, problem-solving, and just fun science!Our students need to be able to apply what they read and learn. If they read a how-to book, they will apply that reading through a hands-on art activity and actually create a product. This is a crucial skill in the real world. By creating and designing their own masterpieces, they are using many critical thinking skills. Students will become more analytical thinkers.
```

In [101]:

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

```
 Creativity is intelligence having fun Albert Einstein Our elementary libr
ary at Greenville Elementary is anything but a quiet hushed space It is a
place for collaboration and research It is a place for incorporating techn
ology It is a place for innovation And it is a place for creating Our scho
ol serves 350 third and fourth graders who primarily live in rural and pov
erty stricken areas in our community Being a Title I school approximately
85 of them receive free or reduced lunch But they are inquisitive creative
and eager to learn They love visiting the library to check out books hear
stories create digital stories and use the computer lab for learning and f
un We want to build our library is Makerspace with activities revolving ar
ound art and literacy to provide more engaging hands on activities We want
to begin Makerspace Fridays Our school recently received a 1000 grant for
books for our arts integrated Makerspace We have received titles such as O
rigami for Everyone How to Make Stuff with Ducktape and Cool Engineering A
ctivities for Girls We now need supplies to correlate with these new infor
mational texts By adding these art and craft supplies students will be abl
e to design and create masterpieces related to their coursework For exampl
e while studying Native Americans students can use the looms and yarn to r
ecreate Navajo and or Pueblo weaving Weaving can also be integrated with l
iteracy through Greek mythology and the story of Arachne Creating art with
perler beads has many possibilities Students can design their own animals
after studying their characteristics They can use symmetry and patterning
to create one of a kind originals Origami reinforces geometry thinking ski
lls fractions problem solving and just fun science Our students need to be
able to apply what they read and learn If they read a how to book they wil
l apply that reading through a hands on art activity and actually create a
product This is a crucial skill in the real world By creating and designin
g their own masterpieces they are using many critical thinking skills Stud
ents will become more analytical thinkers
```

In [102]:

```
# 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 [103]:

```
# 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 [00:47<00:00, 2285.90it/s]
```

In [104]:

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

In [105]:

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

Out[105]:

	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 [106]:

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

In [107]:

```
# 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, 50178.78it/s]
```

In [108]:

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

Out[108]:

```
'empowering students through art learning about then now'
```

In [109]:

```
#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 [110]:

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

Out[110]:

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

In [111]:

```
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 [112]:

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

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)**

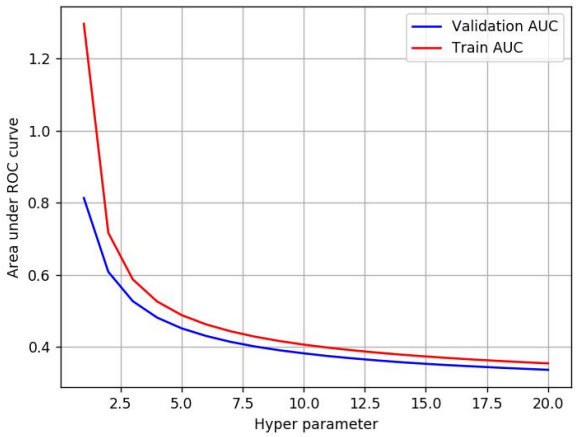
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
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

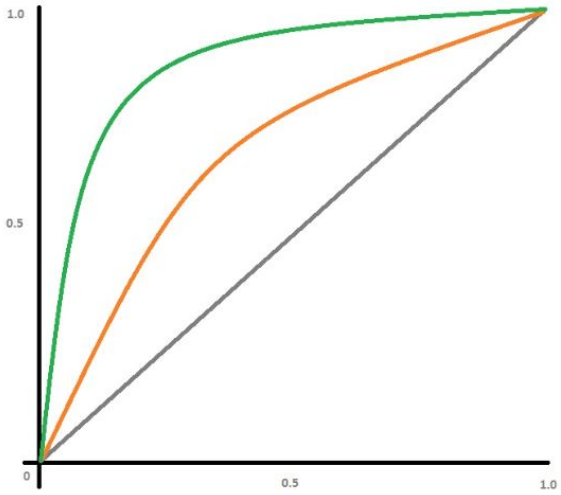
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding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

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

(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 [113]:

```
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 [114]:

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

Out[114]:

```
(109248,)
```

In [115]:

```
#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 [116]:

```
#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 [117]:

```
# 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
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 encoding categorical features

In [118]:

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

Out[118]:

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

In [119]:

```
#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)
vectorizer1.fit(x_train['clean_categories'].values)
#vectorizer.fit(X_train['clean_subcategories'].values)

x_train_categories_one_hot = vectorizer1.transform(x_train['clean_categories'].values)
x_cv_categories_one_hot = vectorizer1.transform(x_cv['clean_categories'].values)
x_test_categories_one_hot = vectorizer1.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 [120]:

```
#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_sub_cat_dict.keys()), lowercase=Fa
lse, binary=True)
vectorizer2.fit(x_train['clean_subcategories'].values)

x_train_subcategories_one_hot = vectorizer2.transform(x_train['clean_subcategories'].va
lues)
x_cv_subcategories_one_hot = vectorizer2.transform(x_cv['clean_subcategories'].values)
x_test_subcategories_one_hot = vectorizer2.transform(x_test['clean_subcategories'].valu
es)
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)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutrition
Education', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'A
ppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Lit
eracy']
Shape of matrix after one hot encodig  (53531, 30)
Shape of matrix after one hot encodig  (22942, 30)
Shape of matrix after one hot encodig  (32775, 30)
```

In [121]:

```
#one hot encoding for school_state


my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())

school_state_cat_dict = dict(my_counter)
sorted_school_state_cat_dict = dict(sorted(school_state_cat_dict.items(), key=lambda kv
: kv[1]))


#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer3 = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()), low
ercase=False, binary=True)
vectorizer3.fit(x_train['school_state'].values)

x_train_school_state_one_hot = vectorizer3.transform(x_train['school_state'].values)
x_cv_school_state_one_hot = vectorizer3.transform(x_cv['school_state'].values)
x_test_school_state_one_hot = vectorizer3.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)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME',
'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'N
V', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'M
A', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'N
Y', 'TX', 'CA']
Shape of matrix after one hot encodig  (53531, 51)
Shape of matrix after one hot encodig  (22942, 51)
Shape of matrix after one hot encodig  (32775, 51)
```

In [122]:

```
#one hot encoding for project_grade_category

my_counter = Counter()
for project_grade in project_data['project_grade_category'].values:
    my_counter.update(project_grade.split())


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




#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer4 = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), lo
wercase=False, binary=True)
vectorizer4.fit(x_train['project_grade_category'].values)

x_train_grade_category_one_hot = vectorizer4.transform(x_train['project_grade_category'
].values)
x_cv_grade_category_one_hot = vectorizer4.transform(x_cv['project_grade_category'].valu
es)
x_test_grade_category_one_hot = vectorizer4.transform(x_test['project_grade_category'].
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)
```

```
['9-12', '6-8', '3-5', 'PreK-2', 'Grades']
Shape of matrix after one hot encodig  (53531, 5)
Shape of matrix after one hot encodig  (22942, 5)
Shape of matrix after one hot encodig  (32775, 5)
```

In [123]:

```
#one hot encoding for project_prefix

my_counter = Counter()
for teacher_prefix in project_data['teacher_prefix'].values:
    teacher_prefix = str(teacher_prefix)
    my_counter.update(teacher_prefix.split())

teacher_prefix_cat_dict = dict(my_counter)
sorted_teacher_prefix_cat_dict = dict(sorted(teacher_prefix_cat_dict.items(), key=lambd
a kv: kv[1]))




#_____________________________
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer5 = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys()), l
owercase=False, binary=True)
vectorizer5.fit(x_train['teacher_prefix'].values)

x_train_prefix_one_hot = vectorizer5.transform(x_train['teacher_prefix'].values)
x_cv_prefix_one_hot = vectorizer5.transform(x_cv['teacher_prefix'].values)
x_test_prefix_one_hot = vectorizer5.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)
```

```
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encodig  (53531, 5)
Shape of matrix after one hot encodig  (22942, 5)
Shape of matrix after one hot encodig  (32775, 5)
```

In [124]:

```
# 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 [125]:

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

Out[125]:

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

In [126]:

```
'''#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[126]:

```
'#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 [127]:

```
'''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[127]:

```
"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 [128]:

```
#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[128]:

```
(53531, 1)
```

In [129]:

```
'''#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[129]:

```
'#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 [130]:

```
'''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[130]:

```
"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 [131]:

```
#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[131]:

```
(22942, 1)
```

In [132]:

```
'''#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[132]:

```
'#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 [133]:

```
'''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[133]:

```
"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 [134]:

```
'''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[134]:

```
"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 [135]:

```
#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[135]:

```
(32775, 1)
```

2.2.3 merge numerical and categorical data

In [136]:

```
# 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, 101)
(22942, 101)
(32775, 101)
```

In [137]:

```
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, 30)
(53531, 51)
(53531, 5)
(53531, 5)
(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 [138]:

```
# 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 [139]:

```
#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 [140]:

```
#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 [141]:

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

merge dataset, SET 1

In [142]:

```
# 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, 14705)
(22942, 14705)
(32775, 14705)
```

In [143]:

```
type(x_train_bow)
```

Out[143]:

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

simple tuning

In [144]:

```
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 tr_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
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

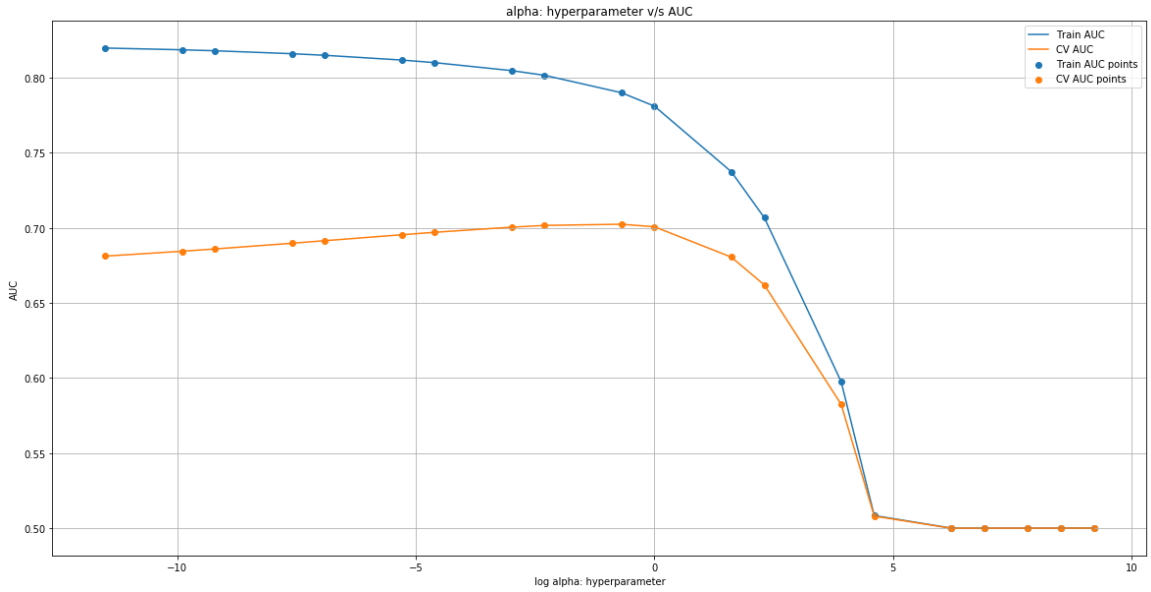
In [145]:

```
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,  7.54it/s]
100%|████████████████████████████████████████████████████████████
████████████████████| 20/20 [00:00<?, ?it/s]
```

In [146]:

```
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 [147]:

```
#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']
```

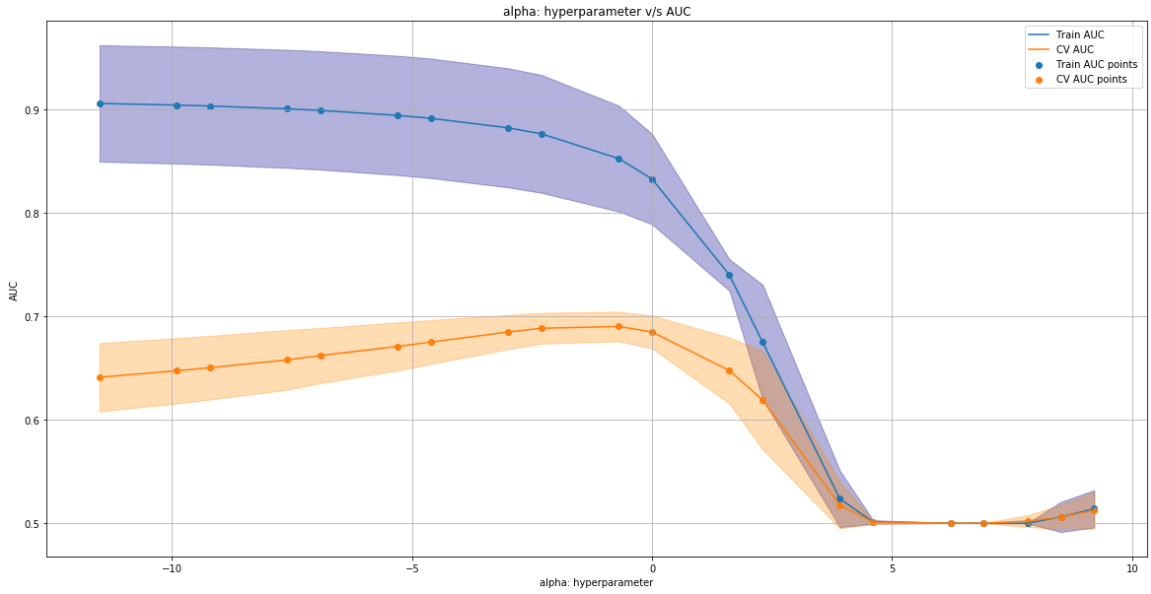
```
GridSearchCV(cv=<generator object TimeSeriesSplit.split at 0x000001D021DB0
E08>,
       error_score='raise',
       estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'alpha': [1e-05, 5e-05, 0.0001, 0.0005, 0.001, 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)
0.690018543840553
0.5
```

optimal alpha value is 0.0001

In [150]:

```
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 [151]:

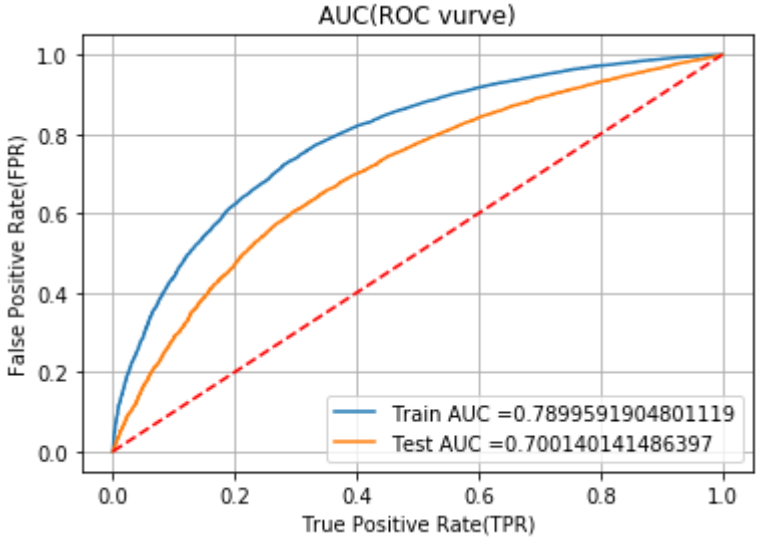
```
alpha_opt_bow=grid.best_estimator_.alpha
```

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 [152]:

```
# 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.plot([0,1],[0,1],'r--')
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC(ROC vurve)")
plt.grid()
plt.show()
```



In [322]:

```
'''# 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()'''
```

Out[322]:

```
"# Plotting the ROC Curve for the Best Classifier\n#_____________________
_____________________________\n#https://datamize.wordpress.com/2015/01/24/how
-to-plot-a-roc-curve-in-scikit-learn/\nfrom sklearn.metrics import roc_cur
ve, auc\nY_score_test = grid.best_estimator_.predict_proba(x_test_bow)\nfp
r1, tpr1, thresholds1 = roc_curve(y_test,Y_score_test[:, 1])\nBOW_roc_auc_
test = auc(fpr1, tpr1)\n\nY_score_train = grid.best_estimator_.predict_pro
ba(x_train_bow)\nfpr2, tpr2, thresholds2 = roc_curve(y_train,Y_score_train
[:, 1])\nBOW_roc_auc_train = auc(fpr2, tpr2)\n\nplt.title('Receiver Operat
ing Characteristic')\nplt.plot(fpr1, tpr1, 'b',label='AUC_test = %0.2f'% B
OW_roc_auc_test)\nplt.plot(fpr2, tpr2, 'g',label='AUC_train = %0.2f'% BOW_
roc_auc_train)\nplt.legend(loc='lower right')\nplt.plot([0,1],[0,1],'r--')
\nplt.xlim([-0.1,1.2])\nplt.ylim([-0.1,1.2])\nplt.ylabel('True Positive Ra
te')\nplt.xlabel('False Positive Rate')\nplt.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 [323]:

```
'''# 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))))'''
```

Out[323]:

```
'# Display Performance of the  Hyper-parametrized BOW model on TEST data\n
\n\ny_pred = grid.best_estimator_.predict(x_test_bow)\n    \n#Evaluate the
model accuracy on TEST data\n\ntest_accuracy_bow = accuracy_score(y_test,
y_pred, normalize=True) * 100\npoints = accuracy_score(y_test, y_pred, nor
malize=False)\n\n# Display the classification report\nprint(classification
_report(y_test, y_pred,digits=4))\n\n#Display the model accuracy on TEST d
ata\nprint(\'\nThe number of accurate predictions out of {} data points on
TEST data is {}\'.format(x_test_bow.shape[0], points))\nprint(\'Accuracy o
f the {} model on TEST data is {} %\'.format("BOW", \'{:f}\'.format(np.rou
nd(test_accuracy_bow,2))))'
```

confusion matrix(test)

In [153]:

```
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
```

Train confusing matrix

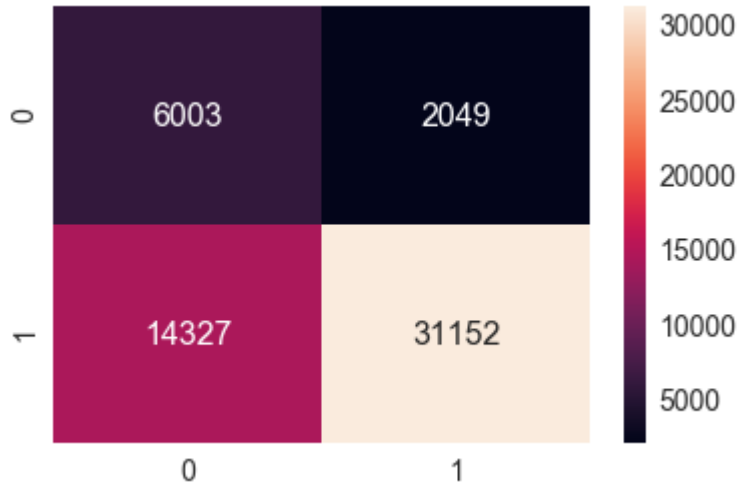
In [154]:

```
print("="*100)
print("TRAIN confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))


conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)), range(2),range(2))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================================
TRAIN confusion matrix
the maximum value of tpr*(1-fpr) 0.5199374649007139 for threshold 0.915
[[ 6003  2049]
 [14327 31152]]
the maximum value of tpr*(1-fpr) 0.5199374649007139 for threshold 0.915
```

Out[154]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d005f41278>
```



TEST confusing matrix

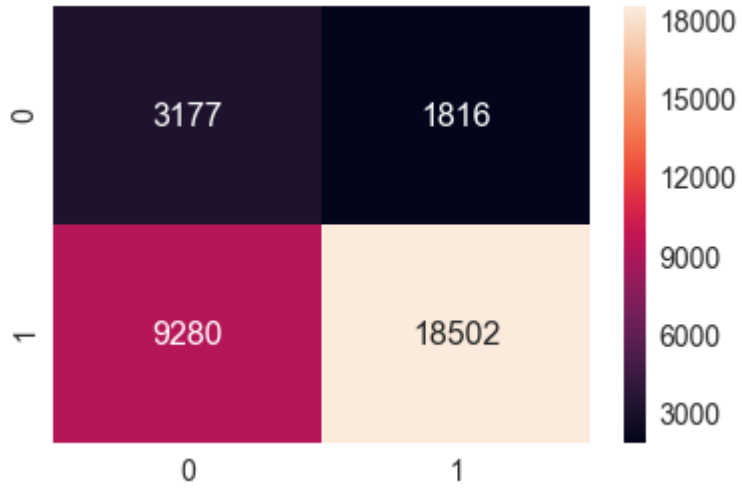
In [155]:

```
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) 0.427542681848029 for threshold 0.907
[[ 3177  1816]
 [ 9280 18502]]
the maximum value of tpr*(1-fpr) 0.427542681848029 for threshold 0.907
```

Out[155]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d00615b1d0>
```



2.4.1.1 Top 10 important features of positive class from SET 1

In [156]:

```
'''#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[156]:

```
'#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 [157]:

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

```
(53531, 14705) (53531,)
(22942, 14705)
(32775, 14705)
```

In [158]:

```
#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[158]:

```
14705
```

In [159]:

```
list(bow_features_probs_neg.values())[-10]
```

Out[159]:

```
-11.261130970737184
```

In [160]:

```
type(bow_features_probs_neg)
```

Out[160]:

```
dict
```

In [161]:

```
#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[161]:

```
100
```

In [162]:

```
# 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[162]:

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

In [163]:

```
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 [164]:

```
len(bow_features_names)
```

Out[164]:

```
14705
```

In [165]:

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

In [166]:

```
#final_bow_features
```

In [167]:

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

In [168]:

```
neg.nunique()
```

Out[168]:

```
feature_prob_estimates      775
feature_names             12648
dtype: int64
```

2.4.1.2 Top 10 important features of negative class from SET 1

In [169]:

```
print('Top 30 negative feature' )
neg.head(30)
```

Top 30 negative feature

Out[169]:

	feature_prob_estimates	feature_names
10829	-3.019891	students
9825	-4.103208	school
6509	-4.423095	learning
2172	-4.588007	classroom
7586	-4.781668	not
6505	-4.783681	learn
5362	-4.816780	help
100	-4.941542	price
94	-4.941542	Grades
7412	-4.988449	nannan
6895	-5.018304	many
7463	-5.104664	need
12382	-5.148136	work
2312	-5.320688	come
6753	-5.363504	love
10232	-5.374165	skills
324	-5.395445	able
6957	-5.397010	materials
9058	-5.416787	reading
2985	-5.420790	day
11902	-5.424205	use
2159	-5.455062	class
12137	-5.476669	want
12459	-5.501359	year
7511	-5.543536	new
6835	-5.547626	make
12422	-5.559304	would
693	-5.589705	also
10828	-5.619102	student
11156	-5.630166	technology

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 [170]:

```
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 [173]:

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

In [174]:

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

In [175]:

```
pos.head(30)
```

Out[175]:

	feature_prob_estimates	feature_names
10829	-3.003392	students
9825	-4.146488	school
6509	-4.512648	learning
2172	-4.533132	classroom
7586	-4.805255	not
6505	-4.846794	learn
5362	-4.875308	help
94	-4.991821	Grades
100	-4.991821	price
6895	-5.018448	many
7412	-5.036862	nannan
7463	-5.147895	need
9058	-5.148487	reading
12382	-5.151268	work
11902	-5.214665	use
6753	-5.307723	love
324	-5.336666	able
2985	-5.338219	day
2312	-5.364905	come
2159	-5.387968	class
12422	-5.416842	would
11156	-5.449176	technology
1479	-5.478075	books
693	-5.486838	also
10232	-5.506141	skills
7511	-5.537329	new
12459	-5.544182	year
6835	-5.586274	make
12137	-5.620645	want
11353	-5.621676	time

2.4.2 Applying Naive Bayes on TFIDF, SET 2

TFIDF Vectorizing essy and title variable, SET 2

In [176]:

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

In [177]:

```
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 [178]:

```
#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 [179]:

```
# 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, 9553)
(22942, 9553)
(32775, 9553)
```

simple tuning

In [180]:

```
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 tr_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
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

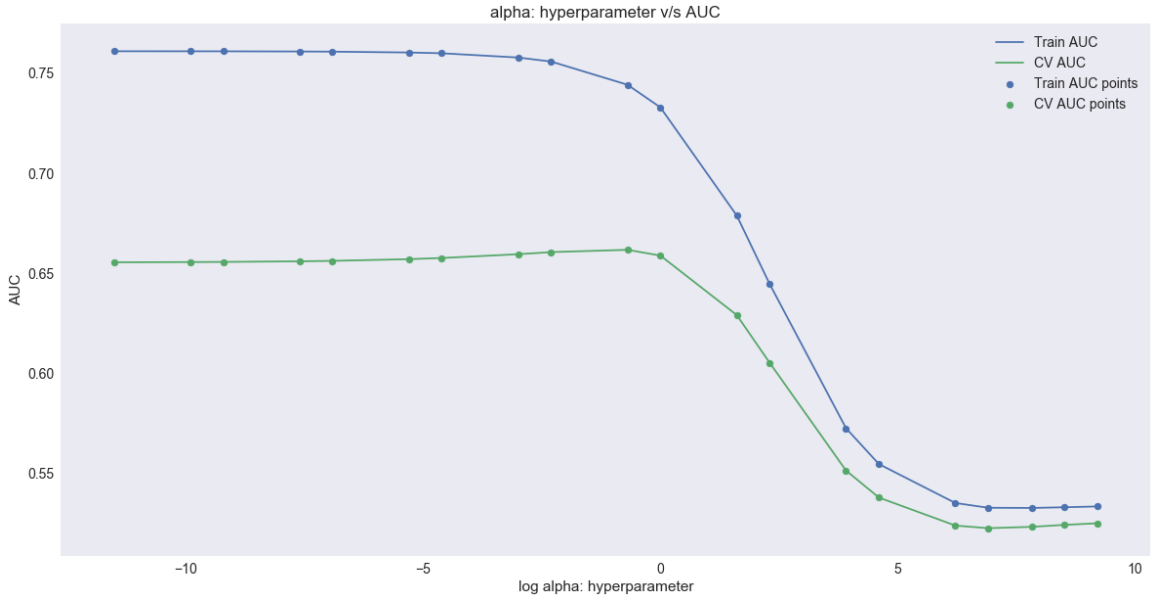
In [181]:

```
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,  4.81it/s]
100%|████████████████████████████████████████████████████████████
████████████████████| 20/20 [00:00<?, ?it/s]
```

In [182]:

```
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 [183]:

```
#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']
```

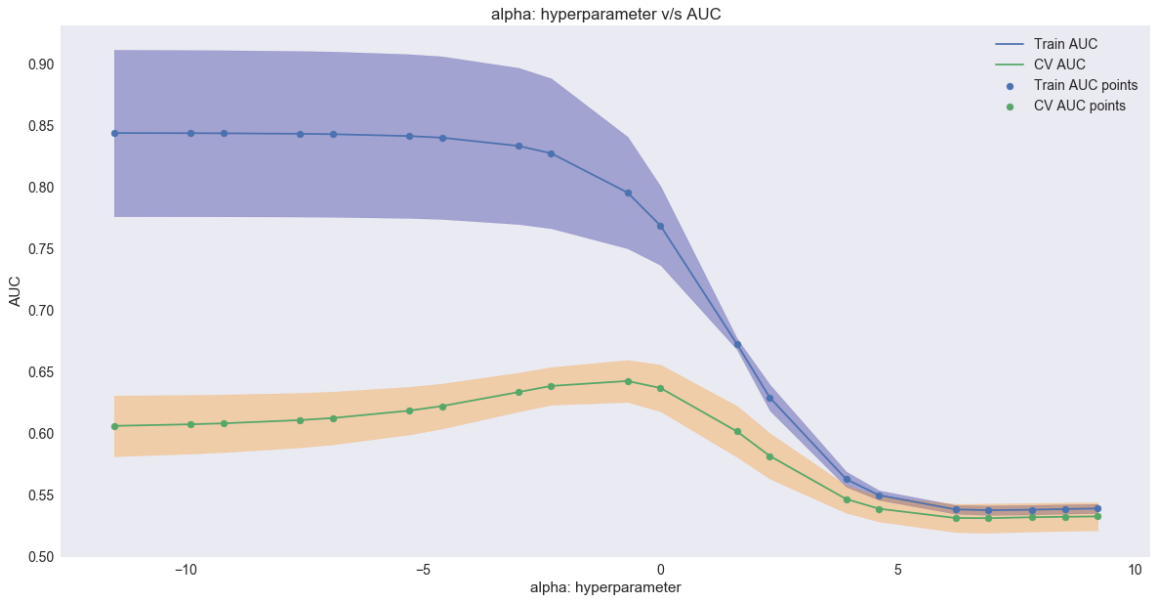
```
GridSearchCV(cv=<generator object TimeSeriesSplit.split at 0x000001D022E5E
E60>,
       error_score='raise',
       estimator=MultinomialNB(alpha=1.0, class_prior=None, fit_prior=Tru
e),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'alpha': [1e-05, 5e-05, 0.0001, 0.0005, 0.001, 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)
0.6425220558210168
0.5
```

optimal value of alpha value in TFIDF train dataset is 1

In [184]:

```
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]
```



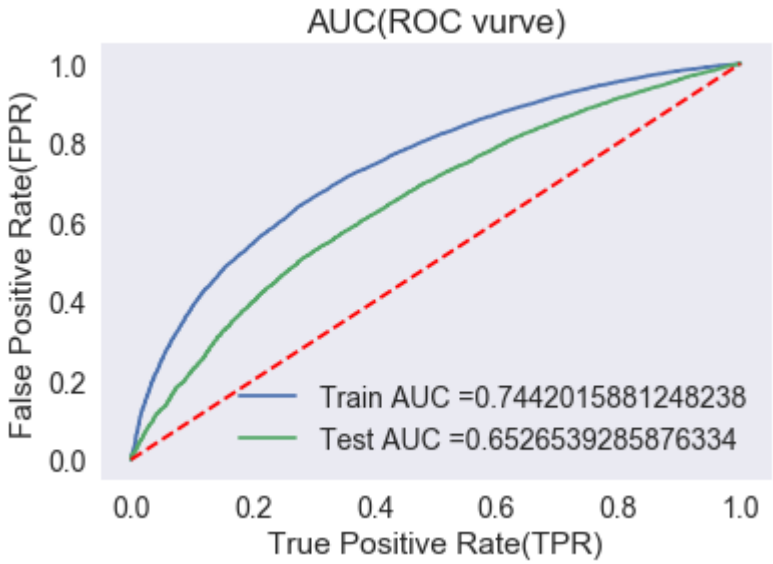
In [185]:

```
alpha_opt_tfidf=grid.best_estimator_.alpha
```

Apply best parameter on test data, SET 2

In [186]:

```
# 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.plot([0,1],[0,1],'r--')
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 [187]:

```
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
```

Training confusion matrix

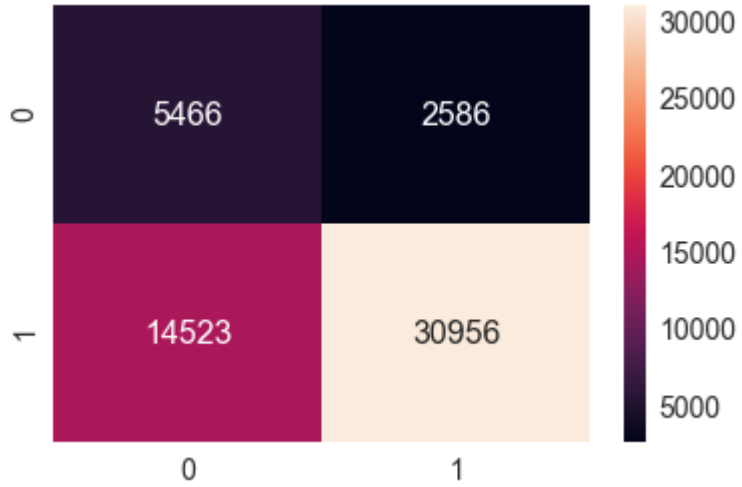
In [188]:

```
print("="*100)
print("TRAIN confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))


conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)), range(2),range(2))

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================================
TRAIN confusion matrix
the maximum value of tpr*(1-fpr) 0.46441388303584485 for threshold 0.845
[[ 5466  2586]
 [14523 30956]]
the maximum value of tpr*(1-fpr) 0.46441388303584485 for threshold 0.845
```

Out[188]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d0c86965f8>
```



Test confusion matrix

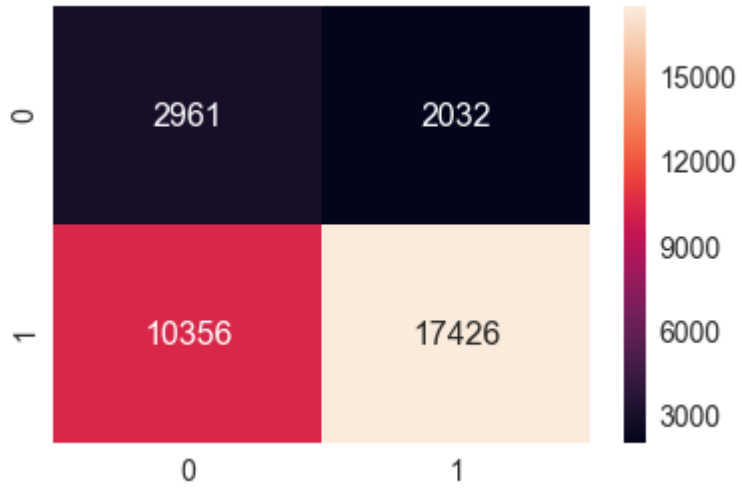
In [189]:

```
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) 0.3755397070692722 for threshold 0.859
[[ 2961  2032]
 [10356 17426]]
the maximum value of tpr*(1-fpr) 0.3755397070692722 for threshold 0.859
```

Out[189]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d02215f470>
```



2.4.2.1 Top 10 important features of negative class from SET 2

In [190]:

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

```
(53531, 9553) (53531,)
(22942, 9553)
(32775, 9553)
```

In [191]:

```
#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[191]:

```
9553
```

In [192]:

```
#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[192]:

```
100
```

In [193]:

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

In [194]:

```
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 [195]:

```
len(tfidf_features_names)
```

Out[195]:

```
9553
```

In [198]:

```
#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 = Fal
se)
```

In [199]:

```
neg.nunique()
```

Out[199]:

```
feature_prob_estimates     669
feature_names             7653
dtype: int64
```

In [200]:

```
print('Top 10 negative feature' )
neg.head(30)
```

Top 10 negative feature

Out[200]:

	feature_prob_estimates	feature_names
6509	-4.423095	future
2172	-4.588007	immigrants
7586	-4.781668	more technology
6505	-4.783681	functional
5362	-4.816780	balancing act
100	-4.941542	price
94	-4.941542	Grades
7412	-4.988449	magic carpet
6895	-5.018304	in
7463	-5.104664	manipulatives
2312	-5.320688	ipads classroom
6753	-5.363504	hear ye hear ye
324	-5.395445	allows students
6957	-5.397010	instrument
9058	-5.416787	toner
2985	-5.420790	necessities
2159	-5.455062	identify
7511	-5.543536	mathematics
6835	-5.547626	hocus pocus
693	-5.589705	certain
7742	-5.717167	news
2076	-5.725835	helping students
6959	-5.743124	integrate
4966	-5.767218	wide
5068	-5.776334	year old students
8	-5.816924	Literacy_Language
8826	-5.833753	tastic
7	-5.864842	Math_Science
8748	-5.865468	styles
3275	-5.912547	percent students

2.4.2.2 Top 10 important features of positive class from SET 2

In [201]:

```
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 [202]:

```
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 = False)
```

In [203]:

```
pos.head(30)
```

Out[203]:

	feature_prob_estimates	feature_names
6509	-4.512648	future
2172	-4.533132	immigrants
7586	-4.805255	more technology
6505	-4.846794	functional
5362	-4.875308	balancing act
94	-4.991821	Grades
100	-4.991821	price
6895	-5.018448	in
7412	-5.036862	magic carpet
7463	-5.147895	manipulatives
9058	-5.148487	toner
6753	-5.307723	hear ye hear ye
324	-5.336666	allows students
2985	-5.338219	necessities
2312	-5.364905	ipads classroom
2159	-5.387968	identify
1479	-5.478075	ensure
693	-5.486838	certain
7511	-5.537329	mathematics
6835	-5.586274	hocus pocus
7742	-5.640469	news
5068	-5.694923	year old students
8	-5.704832	Literacy_Language
671	-5.786246	carpet
6959	-5.790684	integrate
9051	-5.791368	today reader
4966	-5.795928	wide
4127	-5.815928	staff
8826	-5.838693	tastic
2076	-5.840285	helping students

Conclusion

In [204]:

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

In [205]:

```
#!pip install prettytable
```

In [206]:

```
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          | 0.7899591904801119
| 0.700140141486397  |
|   TFIDF    | MultinomialNB |          0.5          | 0.7442015881248238
| 0.6526539285876334 |
+------------+---------------+-----------------------+--------------------
+--------------------+
```

Observation:

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

Test and train accuracy in both case is nearly same. so no case of overfitting or underfitting.

Both case have high AUC value in test and low value in train. it show that model perform well on test dataset.