Assignment 6: Apply NB

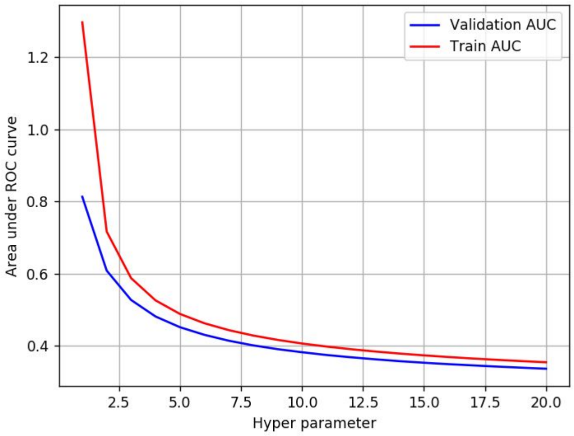
1. Minimum data points need to be considered for people having 4GB RAM is **50k** and for 8GB RAM is **100k**
2. When you are using ramdomsearchcv or gridsearchcv you need not split the data into X_train,X_cv,X_test. As the above methods use kfold. The model will learn better if train data is more so splitting to X_train,X_test will suffice.
3. If you are writing for loops to tune your model then you need split the data into X_train,X_cv,X_test.
4. While splitting the data explore stratify parameter.
5. **Apply Multinomial NB on these feature sets**

    - Features that need to be considered
      essay
      while encoding essay, try to experiment with the max_features and n_grams parameter of vectorizers and see if it increases AUC score.
      categorical features
      - teacher_prefix
      - project_grade_category
      - school_state
      - clean_categories
      - clean_subcategories
      numerical features
      - price
      - teacher_number_of_previously_posted_projects
      while encoding the numerical features check [this (https://imgur.com/ldZA1zg)](https://imgur.com/ldZA1zg) and [this (https://ac-classroom-production.s3.amazonaws.com/public/COMMENT/Annotation_2020-05-21_225912_0lyZzN8.jpg)](https://ac-classroom-production.s3.amazonaws.com/public/COMMENT/Annotation_2020-05-21_225912_0lyZzN8.jpg)

    - Set 1: categorical, numerical features + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)

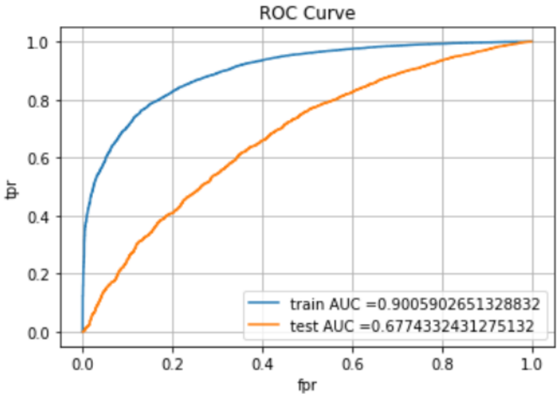
6. **The hyper paramter tuning(find best alpha:smoothing parameter)**

    - Consider alpha values in range: 10^-5 to 10^2 like [0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
    - Explore class_prior = [0.5, 0.5] parameter which can be present in MultinomialNB function(go through [this (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html)](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html)) then check how results might change.
    - Find the best hyper parameter which will give the maximum [AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - For hyper parameter tuning using k-fold cross validation(use GridsearchCV or RandomsearchCV)/simple cross validation data (write for loop to iterate over hyper parameter values)
    - 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

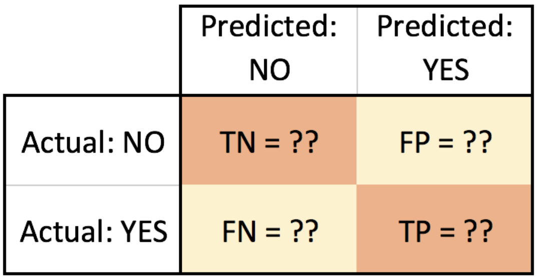


-while plotting take log(alpha) on your X-axis so that it will be more readable

- 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.



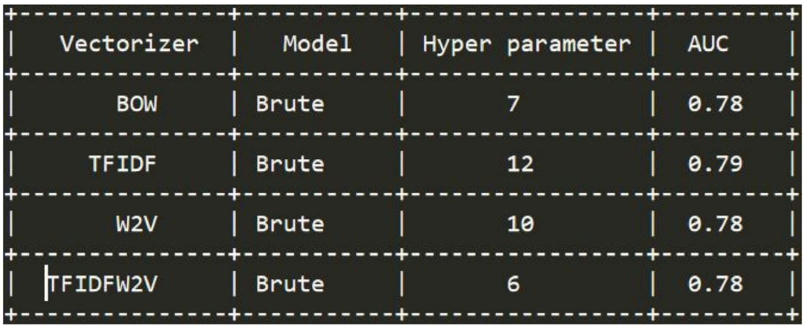
- Along with plotting ROC curve, you need to print the [confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points



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

-plot the confusion matrix in heatmaps, while plotting the confusion matrix go through the [link (https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confusion-matrix-in-exponential-form-to-nor)](https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confusion-matrix-in-exponential-form-to-nor)

7. find the top 20 features from either from feature Set 1 or feature 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 **BOTH** positive as well as negative corresponding feature names.
   - go through the [link (https://imgur.com/mWvE7gj)](https://imgur.com/mWvE7gj)
8. You need to summarize the results at the end of the notebook, summarize it in the table format



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

In [1]:

```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
import string
import matplotlib.pyplot as plt

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

from tqdm import tqdm
import os

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

import sqlite3


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
import re

import string
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
from chart_studio.plotly import plotly
```

In [ ]:

2. Naive Bayes

1.1 Loading Data

In [2]:

```
#make sure you are loading atleast 50k datapoints
#you can work with features of preprocessed_data.csv for the assignment.
# If you want to add more features, you can add. (This is purely optional, not mandatory)
tr_data = pd.read_csv('train_data.csv',nrows= 50000)

resource_data = pd.read_csv('resources.csv')

data = pd.read_csv('preprocessed_data.csv')
```

In [3]:

```
print(tr_data.shape)
print(tr_data.columns)
```

```
(50000, 17)
Index(['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_approve
d'],
      dtype='object')
```

In [4]:

```
print(resource_data.shape)
print(resource_data.columns)
```

```
(1541272, 4)
Index(['id', 'description', 'quantity', 'price'], dtype='object')
```

In [5]:

```
print(data.shape)
print(data.columns)
```

```
(109248, 9)
Index(['school_state', 'teacher_prefix', 'project_grade_category',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price'],
      dtype='object')
```

Preprocessing of project_subject_categories

In [6]:

```
catogories = list(tr_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
list_catg = []
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 the parts
        if 'The' in j.split(): # this will split each of the catogory based on space
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp += j.strip()+" " #" abc ".strip() will return "abc", remove the trailing space
        temp = temp.replace('&','_') # we are replacing the & value into
    list_catg.append(temp.strip())

tr_data['clean_categories'] = list_catg
tr_data.drop(['project_subject_categories'], axis=1, inplace=True)

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

catg_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(catg_dict.items(), key = lambda x: x[1]))
print(sorted_cat_dict)
```

```
{'Warmth': 643, 'Care_Hunger': 643, 'History_Civics': 2689, 'Music_Arts': 46
99, 'AppliedLearning': 5569, 'SpecialNeeds': 6233, 'Health_Sports': 6538, 'M
ath_Science': 18874, 'Literacy_Language': 23998}
```

Preprocessing of project_subject_subcategories

In [7]:

```
sub_catogories = list(tr_data['project_subject_subcategories'].values)

# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_catg_list = []
for i in sub_catogories:
    temp = ""

    for j in i.split(','): # it will split it in the parts
        if 'The' in j.split(): # this will split each of the catogory based on space
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp += j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_catg_list.append(temp.strip())

tr_data['clean_categories'] = sub_catg_list
tr_data.drop(['project_subject_subcategories'], axis=1, inplace=True)


my_counter = Counter()
for word in tr_data['clean_categories'].values:
    my_counter.update(word.split())

sub_catg_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_catg_dict.items(), key=lambda y: y[1]))

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# remove special characters from list of strings python: https://stackoverflow.com/a/473019
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
```

Text preprocessing

In [8]:

```
# merge two column text dataframe:
tr_data["essay"] = tr_data["project_essay_1"].map(str) + tr_data["project_essay_2"].map(str
```

In [9]:

```
# printing some random reviews
print(tr_data['essay'].values[0])
```

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

In [10]:

```
print(tr_data['essay'].values[1000])
print(tr_data['essay'].values[3000])
```

```
How do you remember your days of school? Was it in a sterile environment w
ith plain walls, rows of desks, and a teacher in front of the room? A typi
cal day in our room is nothing like that. I work hard to create a warm inv
iting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed r
aces in Arkansas.\r\nThey attend a Title I school, which means there is a
high enough percentage of free and reduced-price lunch to qualify. Our sch
ool is an \"open classroom\" concept, which is very unique as there are no
walls separating the classrooms. These 9 and 10 year-old students are very
eager learners; they are like sponges, absorbing all the information and e
xperiences and keep on wanting more.With these resources such as the comfy
red throw pillows and the whimsical nautical hanging decor and the blue fi
sh nets, I will be able to help create the mood in our classroom setting t
o be one of a themed nautical environment. Creating a classroom environmen
t is very important in the success in each and every child's education. Th
e nautical photo props will be used with each child as they step foot into
our classroom for the first time on Meet the Teacher evening. I'll take pi
ctures of each child with them, have them developed, and then hung in our
classroom ready for their first day of 4th grade.  This kind gesture will
set the tone before even the first day of school! The nautical thank you c
ards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help m
e to help make our classroom a fun, inviting, learning environment from da
y one.\r\n\r\nIt costs lost of money out of my own pocket on resources to
get our classroom ready. Please consider helping with this project to make
our new school year a very successful one. Thank you!nannan
I am a third grade teacher at Heritage Elementary in Madison Alabama. My s
tudents are unique and each possess special qualities that help others lea
rn and grow. My students and I share experiences that help guide them into
making choices that are productive and mold them into life long learners.
They have an amazing ability to help others achieve their highest academic
potential while ability to help children achieve their best\r\n\r\nI striv
e to inspire my students not only academically but personally. The kids’ e
nergies; their inquisitiveness makes inspires teaching them and pushing th
em harder\r\n\r\n\r\n\r\nFlexible Seating and Student-Centered Classroom R
edesign is key in encouraging students to perform at their highest academi
c potential. My mission is to keep the focus on what's really important: t
he students. If student motivation and higher engagement is truly the desi
red end game, then I must adapt right along with my students in my classro
om.  Our classroom environments should be conducive to open collaboration,
communication, creativity, and critical thinking. This simply cannot be do
ne when kids are sitting in rows of desks all day.\r\n\r\n“Studies on clas
sroom seating suggest that sustained sitting in regular classroom chairs i
s unhealthy for children’s bodies, particularly their backs” (Schilling &S
chwartz, 2004, p. 36).\r\n\r\nAllowing kids some control over where they s
it turns them into problem solvers who can identify how they’re feeling an
d choose what works best for them. Kids simply aren’t meant to sit still a
ll day long…none of us are..\r\n\r\nLastly, these key benefits are essenti
al in promoting a healthy and safe learning environment that can be enjoye
d by ALL students!\r\n\r\n1. promotes students attention spans which resul
ts in higher achievement\r\n2. makes students more actively engaged in the
classroom\r\n3. gives them an active outlet without disrupting their learn
ing\r\n4. makes them more physically fit\r\n5. helps those with ADHD and A
utism, along with other special needs\r\n6. helps develop a sense of commu
nity among the students which improves their social skills\r\n7. helps the
m to become independent learners\r\n8. is LOVED by the students and teache
```

```
r\r\n\r\n\r\nThank you so much for your generous donation!!\r\nMichele Whi
te\r\n\r\nnannan
```

In [11]:

```
print(tr_data['essay'].values[4999])
```

```
Loud and proud are who we are.  We are a special basketball family like no o
ther.  Our school is in a great community with vast diverseness. We are surr
ounded by colleges and low income housing.  We pride ourselves in preparing
our athletes to be great on and off the court.\r\n\r\nOur students recite ev
ery day that, \"We are destined for greatness.\"  I believe this wholehearte
dly.  I am forming winners in life and in basketball.  A great of kids is co
ming your way!We need socks to add to our two uniforms.  Every basketball se
ason our girls basketball team strives to play their best.  Not only do I pu
sh them to give it all on the court I also to teach them to take pride in ho
w they look on the team.  We want to look like a team from head to toe.\r\n
\r\nGirls should feel good about themselves as they play ball and look good
on and off the court.  I have seen lime green socks, purple socks, and all t
he crazy mismatched socks there is.  We need uniformity all the way around.n
annan
```

In [12]:

```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"]
#https://medium.com/@saitejaponugoti/stop-words-in-nlp-5b248dadad47
```

In [13]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [14]:

```
sent = decontracted(tr_data['essay'].values[20000])
print(sent)
```

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

In [15]:

```
# \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)
```

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

In [16]:

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

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

In [ ]:

```

```

In [17]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(tr_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%|███████████████████████████████████████████████████████████████████
████| 50000/50000 [01:43<00:00, 482.20it/s]
```

In [18]:

```
print(tr_data['essay'].values[5000])
```

```
My class is made up of students from various grade levels. We  work hard in
filling learning gaps and have students reach grade level. My students are d
ealing with emotional issues that make it hard for them to handle frustratio
n with tasks and need a lot of individual attention. By learning to work ind
ependently, my students will have the chance to mainstream into other classr
ooms with their peer groups. Our biggest goal with my students is for them t
o learn not only to control their emotions but to learn how to be students.
Many of them have spent a large amount of time absent from school for differ
ent reasons and need to get into the routine of being in class and on task a
ll day. Modeling good classroom routines and task is important for them to m
aster and move back into general education classrooms. Being apart of a Titl
e 1 school means resources that students need are massive and a lot of suppl
ies are shared with parents to make sure homework is completed.Bouncy Bands
will give my students a way to get rid of anxiety, tension, and energy all w
hile staying at their desk and working independently.  Students will use the
bands at either their desk or at a whole group table with a chair and avoid
having to get up or be asked to stop moving. Movement is the key to keeping
students with ADHD and other disabilities focused and finishing up their ass
ignments or staying on task while the teacher is teaching. \r\n       My goa
l is to help my students learn helpful strategies that will allow them to jo
in their peers in the general education setting.   By learning how to mainta
in focus by getting their wiggles and extra energy and grow academically.nan
nan
```

In [19]:

```
# after preprocesing
preprocessed_essays[5000]
```

Out[19]:

```
'class made students various grade levels work hard filling learning gaps st
udents reach grade level students dealing emotional issues make hard handle
frustration tasks need lot individual attention learning work independently
students chance mainstream classrooms peer groups biggest goal students lear
n not control emotions learn students many spent large amount time absent sc
hool different reasons need get routine class task day modeling good classro
om routines task important master move back general education classrooms apa
rt title 1 school means resources students need massive lot supplies shared
parents make sure homework completed bouncy bands give students way get rid
anxiety tension energy staying desk working independently students use bands
either desk whole group table chair avoid get asked stop moving movement key
keeping students adhd disabilities focused finishing assignments staying tas
k teacher teaching goal help students learn helpful strategies allow join pe
ers general education setting learning maintain focus getting wiggles extra
energy grow academically nannan'
```

In [20]:

```
# after preprocesing
tr_data['preprocessed_essays'] = preprocessed_essays
```

In [21]:

```
print(tr_data['preprocessed_essays'].values[5000])
```

```
class made students various grade levels work hard filling learning gaps stu
dents reach grade level students dealing emotional issues make hard handle f
rustration tasks need lot individual attention learning work independently s
tudents chance mainstream classrooms peer groups biggest goal students learn
not control emotions learn students many spent large amount time absent scho
ol different reasons need get routine class task day modeling good classroom
routines task important master move back general education classrooms apart
title 1 school means resources students need massive lot supplies shared par
ents make sure homework completed bouncy bands give students way get rid anx
iety tension energy staying desk working independently students use bands ei
ther desk whole group table chair avoid get asked stop moving movement key k
eeping students adhd disabilities focused finishing assignments staying task
teacher teaching goal help students learn helpful strategies allow join peer
s general education setting learning maintain focus getting wiggles extra en
ergy grow academically nannan
```

Preprocessing of project_title

In [22]:

```
# similarly you can preprocess the titles also

# printing some random titles.
print(tr_data['project_title'].values[0])
print(tr_data['project_title'].values[150])
print(tr_data['project_title'].values[200])
print(tr_data['project_title'].values[500])
```

```
Educational Support for English Learners at Home
More Movement with Hokki Stools
Sensory toys to make sense of our world
Classroom Chromebooks for College Bound Seniors!
```

In [23]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(tr_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(a for a in sent.split() if a not in stopwords)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|███████████████████████████████████████████████████████████████████████
███| 50000/50000 [00:05<00:00, 9067.47it/s]
```

In [24]:

```
# after preprocesing
preprocessed_title[100]
```

Out[24]:

```
'21st century learners 21st century technology'
```

In [25]:

```
print(tr_data['project_title'].values[0])

print(tr_data['project_title'].values[150])

print(tr_data['project_title'].values[200])

print(tr_data['project_title'].values[400])
```

```
Educational Support for English Learners at Home
More Movement with Hokki Stools
Sensory toys to make sense of our world
Leave It Better Than You Found It!
```

In [26]:

```
tr_data['preprocessed_title'] = preprocessed_title
```

In [27]:

```
print(tr_data['preprocessed_title'].values[0])
print(tr_data['preprocessed_title'].values[150])
```

```
educational support english learners home
more movement hokki stools
```

In [28]:

```
print(tr_data['preprocessed_title'].values[200])
print(tr_data['preprocessed_title'].values[500])
```

```
sensory toys make sense world
classroom chromebooks college bound seniors
```

Categorical data Processing

In [29]:

```
#Project Grade Category
tr_data['project_grade_category'].value_counts()
```

Out[29]:

```
Grades PreK-2    20316
Grades 3-5       16968
Grades 6-8        7750
Grades 9-12       4966
Name: project_grade_category, dtype: int64
```

In [30]:

```
# we need to remove spaces, replace '-' with '_' and conver all letters to small
tr_data['project_grade_category'] = tr_data['project_grade_category'].str.lower()
tr_data['project_grade_category'] = tr_data['project_grade_category'].str.replace(' ','_')
tr_data['project_grade_category'].value_counts()
tr_data['project_grade_category'] = tr_data['project_grade_category'].str.replace('-','_')
```

In [31]:

```
#teacher_prefix
print(tr_data['teacher_prefix'].value_counts())
print(tr_data['teacher_prefix'].isnull().values.any())
print("Nan Values",tr_data['teacher_prefix'].isnull().values.sum())
```

```
Mrs.       26140
Ms.        17936
Mr.         4859
Teacher     1061
Dr.            2
Name: teacher_prefix, dtype: int64
True
Nan Values 2
```

In [32]:

```
tr_data['teacher_prefix'] = tr_data['teacher_prefix'].fillna('Mrs.')
tr_data['teacher_prefix'].value_counts()
```

Out[32]:

```
Mrs.       26142
Ms.        17936
Mr.         4859
Teacher     1061
Dr.            2
Name: teacher_prefix, dtype: int64
```

In [33]:

```
#remove '.' and convert chars to small
tr_data['teacher_prefix'] = tr_data['teacher_prefix'].str.replace('.','')
tr_data['teacher_prefix'] = tr_data['teacher_prefix'].str.lower()
```

In [34]:

```
tr_data['teacher_prefix'].value_counts()
```

Out[34]:

```
mrs        26142
ms         17936
mr          4859
teacher     1061
dr             2
Name: teacher_prefix, dtype: int64
```

In [35]:

```
#school_state
tr_data['school_state'].value_counts()
```

Out[35]:

```
CA    7024
NY    3393
TX    3320
FL    2839
NC    2340
IL    1967
SC    1830
GA    1828
MI    1468
PA    1419
OH    1180
IN    1171
MO    1166
WA    1103
LA    1094
MA    1076
OK    1074
NJ    1005
AZ     994
VA     916
WI     833
UT     792
AL     790
TN     774
CT     774
MD     668
NV     665
KY     614
MS     598
OR     577
MN     556
CO     538
AR     446
IA     306
ID     302
KS     285
DC     247
HI     239
NM     236
ME     222
WV     218
DE     155
AK     153
NE     144
SD     142
NH     141
RI     126
MT     106
ND      63
WY      51
VT      32
Name: school_state, dtype: int64
```

In [36]:

```
tr_data['school_state'] = tr_data['school_state'].str.lower()
tr_data['school_state'].value_counts()
```

Out[36]:

```
ca    7024
ny    3393
tx    3320
fl    2839
nc    2340
il    1967
sc    1830
ga    1828
mi    1468
pa    1419
oh    1180
in    1171
mo    1166
wa    1103
la    1094
ma    1076
ok    1074
nj    1005
az     994
va     916
wi     833
ut     792
al     790
tn     774
ct     774
md     668
nv     665
ky     614
ms     598
or     577
mn     556
co     538
ar     446
ia     306
id     302
ks     285
dc     247
hi     239
nm     236
me     222
wv     218
de     155
ak     153
ne     144
sd     142
nh     141
ri     126
mt     106
nd      63
wy      51
vt      32
Name: school_state, dtype: int64
```

In [37]:

```
re_data = pd.read_csv("C:\\Users\\visha\\AAIC\\Assignment_8\\resources.csv")
re_data.head(3)
```

Out[37]:

	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
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45

In [38]:

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

Out[38]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [39]:

```
# join two dataframes in python:
tr_data = pd.merge(tr_data, price_data, on='id', how='left')
tr_data.head(2)
```

Out[39]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs	in	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr	fl	

2 rows × 21 columns

Type *Markdown* and LaTeX: α^2

we are going to consider

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

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

In [40]:

```
# write your code in following steps for task 1
# 1. Split your data.
X = tr_data.drop(['project_is_approved'], axis=1)
X.head(2)
```

Out[40]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs	in	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr	fl	

In [41]:

```
y = tr_data['project_is_approved'].values
```

In [42]:

```
# Split the dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
# 1) If you want to apply simple cross-validation, split the dataset into 3 parts (ie., tra
# 2) If you want to apply K-fold CV (or) GridSearch Cross Validation (or) Randomized Search
```

1.3 Make Data Model Ready: encoding essay, and project_title

Selecting top 2000 words from essay and project_title

In [ ]:

In [43]:

```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

# Apply Bag of Words (BOW) vectorization on 'Preprocessed_Essay'
# Apply Bag of Words (BOW) vectorization on 'Preprocessed_Title' (Optional)
```

```
(22445, 20) (22445,)
(11055, 20) (11055,)
(16500, 20) (16500,)
```

In [44]:

```
#Categorical Feature
# from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_essay = CountVectorizer(min_df=10,max_features = 5000,ngram_range=(1,5))
vectorizer_essay.fit(X_train['essay'].values)
```

Out[44]:

```
CountVectorizer(max_features=5000, min_df=10, ngram_range=(1, 5))
```

In [45]:

```
X_train_essay_Set1 = vectorizer_essay.transform(X_train['essay'].values)
X_test_essay_Set1 = vectorizer_essay.transform(X_test['essay'].values)
X_cv_essay_Set1 = vectorizer_essay.transform(X_cv['essay'].values)
```

In [46]:

```
print(X_train_essay_Set1.shape,y_train.shape)
print(X_test_essay_Set1.shape,y_test.shape)
print(X_cv_essay_Set1.shape,y_cv.shape)
```

```
(22445, 5000) (22445,)
(16500, 5000) (16500,)
(11055, 5000) (11055,)
```

In [47]:

```
# Apply TF-IDF vectorization on 'Preprocessed_Essay'
# Apply TF-IDF vectorization on 'Preprocessed_Title' (Optional)
from sklearn.feature_extraction.text import TfidfVectorizer
TfidfVectorizer = TfidfVectorizer(min_df=10,max_features = 5000)
TfidfVectorizer.fit(X_train['essay'].values)
```

Out[47]:

```
TfidfVectorizer(max_features=5000, min_df=10)
```

In [48]:

```
X_train_tfidf_Set2 = TfidfVectorizer.transform(X_train['essay'].values)
X_test_tfidf_Set2 = TfidfVectorizer.transform(X_test['essay'].values)
X_cv_tfidf_Set2 = TfidfVectorizer.transform(X_cv['essay'].values)
```

In [49]:

```
print(X_train_tfidf_Set2.shape,y_train.shape)
print(X_test_tfidf_Set2.shape,y_test.shape)
print(X_cv_tfidf_Set2.shape,y_cv.shape)
```

```
(22445, 5000) (22445,)
(16500, 5000) (16500,)
(11055, 5000) (11055,)
```

1.4 Make Data Model Ready: encoding numerical, categorical features

Vectorizing Categorical features

In [50]:

```
# Vectorizing school_state
#code ref: https://www.youtube.com/watch?time_continue=849&v=ZhLXULFjIjQ&feature=emb_logo
#provided did the preprocessing on school state feature
vectorizer_school_state = CountVectorizer(binary=True)
vectorizer_school_state.fit(X_train['school_state'].values)
```

Out[50]:

```
CountVectorizer(binary=True)
```

In [51]:

```
X_train_school_state = vectorizer_school_state.transform(X_train['school_state'].values)
X_cv_school_state = vectorizer_school_state.transform(X_cv['school_state'].values)
X_test_school_state = vectorizer_school_state.transform(X_test['school_state'].values)
```

In [52]:

```


print(X_train_school_state.shape, y_train.shape)
print(X_cv_school_state.shape, y_cv.shape)
print(X_test_school_state.shape, y_test.shape)
```

```
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
```

In [53]:

```
#teacher_prefix encoding
vectorizer_teacher_prefix = CountVectorizer(binary=True)
vectorizer_teacher_prefix.fit(X_train['teacher_prefix'].values)
```

Out[53]:

```
CountVectorizer(binary=True)
```

In [54]:

```
# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_prefix = vectorizer_teacher_prefix.transform(X_train['teacher_prefix'].valu
X_cv_teacher_prefix = vectorizer_teacher_prefix.transform(X_cv['teacher_prefix'].values)
X_test_teacher_prefix = vectorizer_teacher_prefix.transform(X_test['teacher_prefix'].values
```

In [55]:

```
print(X_train_teacher_prefix.shape, y_train.shape)
print(X_cv_teacher_prefix.shape, y_cv.shape)
print(X_test_teacher_prefix.shape, y_test.shape)
```

```
(22445, 5) (22445,)
(11055, 5) (11055,)
(16500, 5) (16500,)
```

In [56]:

```
#Vectorizing : clean_categories

# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_clean_categories = CountVectorizer(lowercase = False, binary = True)
vectorizer_clean_categories.fit(X_train['clean_categories'].values)
print(vectorizer_clean_categories.get_feature_names())
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Governmen
t', 'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Ec
onomics', 'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'F
oreignLanguages', 'Gym_Fitness', 'Health_LifeScience', 'Health_Wellness', 'H
istory_Geography', 'Literacy', 'Literature_Writing', 'Mathematics', 'Music',
'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
```

In [57]:

```
X_train_categories = vectorizer_clean_categories.transform(X_train['clean_categories'].valu
X_cv_categories = vectorizer_clean_categories.transform(X_cv['clean_categories'].values)
X_test_categories = vectorizer_clean_categories.transform(X_test['clean_categories'].values
```

In [58]:

```


print(X_train_categories.shape, y_train.shape)
print(X_cv_categories.shape, y_cv.shape)
print(X_test_categories.shape, y_test.shape)
```

```
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
```

In [59]:

```
#encoding project_grade_category

vectorizer_project_grade_category = CountVectorizer(binary=True)
vectorizer_project_grade_category.fit(X_train['project_grade_category'].values)
```

Out[59]:

```
CountVectorizer(binary=True)
```

In [60]:

```
# we use the fitted CountVectorizer to convert the text to vector
X_train_project_grade_category = vectorizer_project_grade_category.transform(X_train['proje
X_cv_project_grade_category = vectorizer_project_grade_category.transform(X_cv['project_gra
X_test_project_grade_category = vectorizer_project_grade_category.transform(X_test['project
```

In [61]:

```
print(X_train_project_grade_category.shape, y_train.shape)
print(X_cv_project_grade_category.shape, y_cv.shape)
print(X_test_project_grade_category.shape, y_test.shape)
```

```
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
```

In [62]:

```
print(X_train.columns)
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'essay', 'preprocessed_essays', 'preprocessed_title', 'price',
       'quantity'],
      dtype='object')
```

In [63]:

```
from sklearn.preprocessing import Normalizer
normalizer_1 = Normalizer()
```

In [64]:

```
normalizer_1.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,
```

Out[64]:

```
Normalizer()
```

In [65]:

```
X_train_norm = normalizer_1.transform(X_train['teacher_number_of_previously_posted_projects
X_cv_norm = normalizer_1.transform(X_cv['teacher_number_of_previously_posted_projects'].val
X_test_norm = normalizer_1.transform(X_test['teacher_number_of_previously_posted_projects']
```

In [66]:

```
print(X_train_norm.shape, y_train.shape)
print(X_cv_norm.shape, y_cv.shape)
print(X_test_norm.shape, y_test.shape)
```

```
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
```

In [67]:

```
from sklearn.preprocessing import Normalizer
normalizer_2 = Normalizer()
```

In [68]:

```
normalizer_2.fit(X_train['price'].values.reshape(-1,1))
```

Out[68]:

```
Normalizer()
```

In [69]:

```
X_train_norm_price = normalizer_2.transform(X_train['price'].values.reshape(-1,1))
X_cv_norm_price = normalizer_2.transform(X_cv['price'].values.reshape(-1,1))
X_test_norm_price = normalizer_2.transform(X_test['price'].values.reshape(-1,1))
```

In [70]:

```
print(X_train_norm_price.shape, y_train.shape)
print(X_cv_norm_price.shape, y_cv.shape)
print(X_test_norm_price.shape, y_test.shape)
```

```
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
```

Encoding : preprocessed_essays

In [71]:

```
vectorizer_preprocessed_essays = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=
vectorizer_preprocessed_essays.fit(X_train['preprocessed_essays'].values)
```

Out[71]:

```
CountVectorizer(max_features=5000, min_df=10, ngram_range=(1, 4))
```

In [72]:

```
# we use the fitted CountVectorizer to convert the text to vector
X_train_essay = vectorizer_preprocessed_essays.transform(X_train['preprocessed_essays'].val
X_cv_essay = vectorizer_preprocessed_essays.transform(X_cv['preprocessed_essays'].values)
X_test_essay = vectorizer_preprocessed_essays.transform(X_test['preprocessed_essays'].value
```

In [73]:

```
print(X_train_essay.shape, y_train.shape)
print(X_test_essay.shape, y_cv.shape)
print(X_cv_essay.shape, y_test.shape)
```

```
(22445, 5000) (22445,)
(16500, 5000) (11055,)
(11055, 5000) (16500,)
```

Encoding : preprocessed_title

In [74]:

```
vectorizer_preprocessed_title = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5
vectorizer_preprocessed_title.fit(X_train['preprocessed_title'].values)
```

Out[74]:

```
CountVectorizer(max_features=5000, min_df=10, ngram_range=(1, 4))
```

In [75]:

```
X_train_title = vectorizer_preprocessed_title.transform(X_train['preprocessed_title'].value
X_cv_title = vectorizer_preprocessed_title.transform(X_cv['preprocessed_title'].values)
X_test_title = vectorizer_preprocessed_title.transform(X_test['preprocessed_title'].values)
```

In [76]:

```
print(X_train_title.shape, y_train.shape)
print(X_test_title.shape, y_test.shape)
print(X_cv_title.shape, y_cv.shape)
```

```
(22445, 1971) (22445,)
(16500, 1971) (16500,)
(11055, 1971) (11055,)
```

In [77]:

```
print(len(X_train))

X_train.columns
```

```
22445
```

Out[77]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'essay', 'preprocessed_essays', 'preprocessed_title', 'price',
       'quantity'],
      dtype='object')
```

In [78]:

```
from sklearn.feature_extraction.text import CountVectorizer
project_resource_summary = CountVectorizer(min_df=5,tokenizer = lambda x: x.split(), max_fe
project_resource_summary.fit(X_train['project_resource_summary']) # fit has to happen only
```

Out[78]:

```
CountVectorizer(max_features=2000, min_df=5, ngram_range=(1, 4),
                tokenizer=<function <lambda> at 0x000001EF938293A0>)
```

In [79]:

```
X_train_summary_bow = project_resource_summary.transform(X_train['project_resource_summary'
X_cv_summary_bow = project_resource_summary.transform(X_cv['project_resource_summary'])
X_test_summary_bow = project_resource_summary.transform(X_test['project_resource_summary'])
```

In [80]:

```
print(X_train_summary_bow.shape, y_train.shape)
print(X_cv_summary_bow.shape, y_cv.shape)
print(X_test_summary_bow.shape, y_test.shape)
```

```
(22445, 2000) (22445,)
(11055, 2000) (11055,)
(16500, 2000) (16500,)
```

In [81]:

```
from scipy.sparse import hstack
X_train_set1 = hstack((X_train_title, X_train_essay, X_train_categories, X_train_school_sta
X_cv_set1= hstack((X_cv_title, X_cv_essay, X_cv_categories, X_cv_school_state, X_cv_teacher
X_test_set1 = hstack((X_test_title, X_test_essay, X_test_categories, X_test_school_state, X
```

In [82]:

```
print(X_train_set1.shape, y_train.shape)
print(X_cv_set1.shape, y_cv.shape)
print(X_test_set1.shape, y_test.shape)
```

```
(22445, 7063) (22445,)
(11055, 7063) (11055,)
(16500, 7063) (16500,)
```

1.5 Appling NB on different kind of featurization as mentioned in the instructions

Apply NB on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instructions

Set 1

In [83]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    pred = []
    loop_of_train = data.shape[0] - data.shape[0]%1000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, loop_of_train, 1000):
        pred.extend(clf.predict_proba(data[i:i+1000])[:,1])

    if data.shape[0]%1000 !=0:
        pred.extend(clf.predict_proba(data[loop_of_train:])[:,1])

    return pred
```

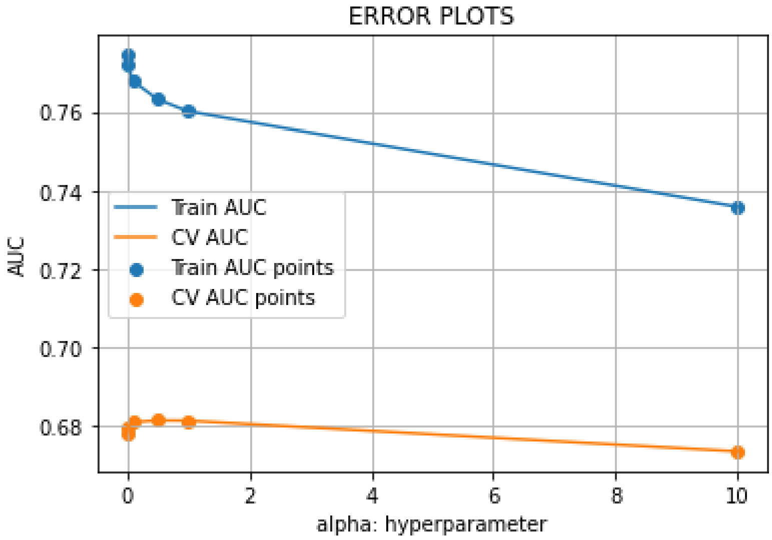
In [84]:

```
# Perform Hyperparameter Tuning.

import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
AUC_of_train = []
AUC_of_cv = []
alpha = [0.001, 0.01, 0.1, 0.5, 1.0, 10.0,]
for i in tqdm(alpha):
    model = MultinomialNB(alpha=i)
    model.fit(X_train_set1, y_train)

    y_pred_train = batch_predict(model, X_train_set1)
    y_pred_cv = batch_predict(model, X_cv_set1)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    AUC_of_train.append(roc_auc_score(y_train,y_pred_train))
    AUC_of_cv.append(roc_auc_score(y_cv, y_pred_cv))
```

```
100%|████████████████████████████████████████████████████████████████████
████████████| 6/6 [00:02<00:00,  2.29it/s]
```

In [85]:

```
plt.plot(alpha, AUC_of_train, label='Train AUC')
plt.plot(alpha, AUC_of_cv, label='CV AUC')

plt.scatter(alpha, AUC_of_train, label='Train AUC points')
plt.scatter(alpha, AUC_of_cv, label='CV AUC points')

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

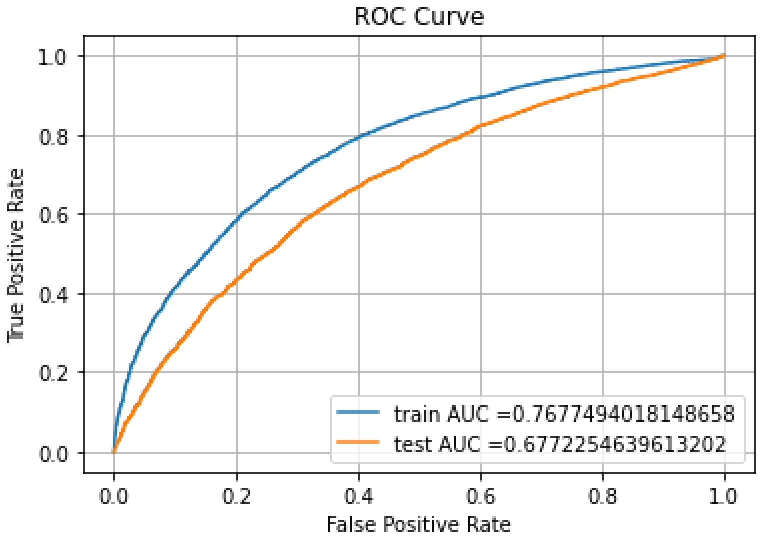


In [86]:

```
# Obtain the optimal value for 'alpha' and using the obtained optimal 'alpha' value, fit a
# Note: If you have split the datase into 3 parts (ie., train, cv and test sets) in the beg
# Make class label and probability predictions on the train and test data.

#here we are choosing best value of alpha based on for loop results
best_alpha = 0.1
```

In [87]:

```
from sklearn.metrics import roc_curve, auc


classifier = MultinomialNB(alpha = best_alpha)
classifier.fit(X_train_set1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_pred_train = batch_predict(classifier, X_train_set1)
y_pred_test = batch_predict(classifier, X_test_set1)

FPR_train, TPR_train, train_thresholds = roc_curve(y_train, y_pred_train)
FPR_test, TPR_test, test_thresholds = roc_curve(y_test, y_pred_test)



plt.plot(FPR_train, TPR_train, label="train AUC ="+str(auc(FPR_train, TPR_train)))
plt.plot(FPR_test, TPR_test, label="test AUC ="+str(auc(FPR_test, TPR_test)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.grid()
plt.show()
```

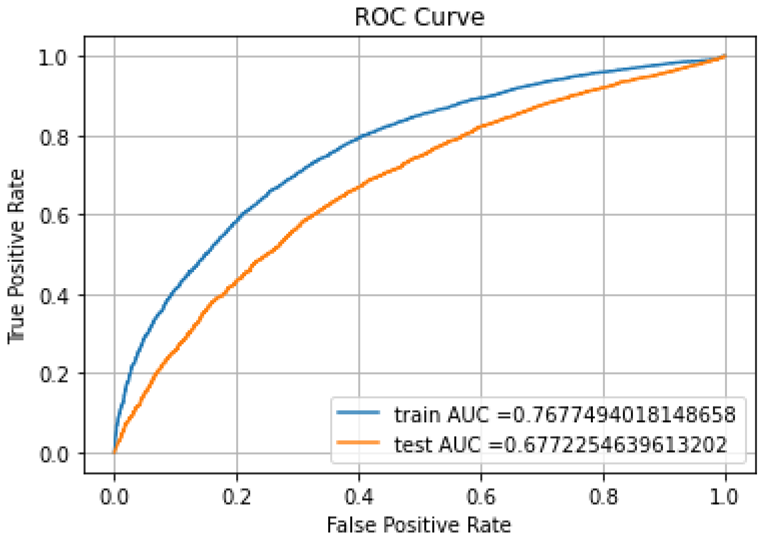


In [88]:

```
# Plot the ROC-AUC curves using the probability predictions made on train and test data.
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.
from sklearn.metrics import roc_curve, auc


classifier = MultinomialNB(alpha = best_alpha)
classifier.fit(X_train_set1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_pred_train = batch_predict(classifier, X_train_set1)
y_pred_test = batch_predict(classifier, X_test_set1)

FPR_train, TPR_train, train_thresholds = roc_curve(y_train, y_pred_train)
FPR_test, TPR_test, test_thresholds = roc_curve(y_test, y_pred_test)
```

In [89]:

```

plt.plot(FPR_train, TPR_train, label="train AUC ="+str(auc(FPR_train, TPR_train)))
plt.plot(FPR_test, TPR_test, label="test AUC ="+str(auc(FPR_test, TPR_test)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.grid()
plt.show()
```



In [90]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def thresholds_(threshould, FPR, TPR):
    t = threshould[np.argmax(TPR*(1-FPR))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("The Maximum value of TPR*(1-FPR)", max(TPR*(1-FPR)), "for threshold", np.round(t
    return t
```

In [91]:

```
def best_t_prediction(proba, threshould):
    predictions = []
    for i in proba:
        if i >= threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [92]:

```

#ref link: http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
from sklearn.metrics import confusion_matrix
best_t = thresholds_(train_thresholds, FPR_train, TPR_train)
print("Train confusion matrix")
print(confusion_matrix(y_train, best_t_prediction(y_pred_train, best_t)))
tn,fp,fn,tp = confusion_matrix(y_train, best_t_prediction(y_pred_train, best_t)).ravel()
```

```
The Maximum value of TPR*(1-FPR) 0.4938389129413086 for threshold 0.825
Train confusion matrix
[[ 2432  1031]
 [ 5634 13348]]
```

In [93]:

```
PT = PrettyTable()
PT.title= 'Train confusion matrix'
PT.field_names= ("","Predicted: NO" ,"Predicted: YES")
PT.add_row(["Actual : No",tn,fp ])
PT.add_row(["Actual : Yes", fn,tp])
print(PT)

print(confusion_matrix(y_test, best_t_prediction(y_pred_test, best_t)))
tn,fp,fn,tp = confusion_matrix(y_test, best_t_prediction(y_pred_test, best_t)).ravel()
```

```
+-----------------------------------------------+
|             Train confusion matrix            |
+--------------+---------------+----------------+
|              | Predicted: NO | Predicted: YES |
+--------------+---------------+----------------+
| Actual : No  |      2432     |      1031      |
| Actual : Yes |      5634     |     13348      |
+--------------+---------------+----------------+
[[1476 1070]
 [4351 9603]]
```

In [94]:

```
PT = PrettyTable()
PT.title= 'Test confusion matrix'
PT.field_names= ("","Predicted: No" ,"Predicted: Yes")
PT.add_row(["Actual : No",tn,fp ])
PT.add_row(["Actual : Yes", fn,tp])
print(PT)
```

```
+-----------------------------------------------+
|             Test confusion matrix             |
+--------------+---------------+----------------+
|              | Predicted: No | Predicted: Yes |
+--------------+---------------+----------------+
| Actual : No  |      1476     |      1070      |
| Actual : Yes |      4351     |      9603      |
+--------------+---------------+----------------+
```

In [95]:

```
# Get feature names from Set1
feature_names_of_set1=[]


for i in vectorizer_project_grade_category.get_feature_names(): #Get features correspoindin
    feature_names_of_set1.append(i)


for i in vectorizer_clean_categories.get_feature_names(): #Get features correspoind to clea
    feature_names_of_set1.append(i)


for i in vectorizer_teacher_prefix.get_feature_names(): #Get features correspoinding to tea
    feature_names_of_set1.append(i)


for i in vectorizer_preprocessed_title.get_feature_names(): #Get features correspoind to pr
    feature_names_of_set1.append(i)


for i in vectorizer_school_state.get_feature_names(): #Get features correspoinding to schoo
    feature_names_of_set1.append(i)


for i in vectorizer_preprocessed_essays.get_feature_names(): #Get features correspoind to p
    feature_names_of_set1.append(i)

feature_names_of_set1.append("teacher_number_of_previously_posted_projects")
feature_names_of_set1.append("price")

print(len(feature_names_of_set1))
```

```
7063
```

In [96]:

```
# Pick the best threshold among the probability estimates, such that it has to yield maximu
# Plot the confusion matrices(each for train and test data) afer encoding the predicted cla
# Get top 20 features displayed for both the classes


sorted_negative_class = (-model.feature_log_prob_[0, :]).argsort()
sorted_positive_class = (-model.feature_log_prob_[1, :]).argsort()

feature_of_negative_class = np.take(feature_names_of_set1, sorted_negative_class[:20])
feature_of_positive_class = np.take(feature_names_of_set1, sorted_positive_class[:20])

print("\nThe top 20 features from the negative class are :\n")
print(feature_of_negative_class, sorted_negative_class[:20])

print("The top 20 features from the positive class are:\n")
print(feature_of_positive_class, sorted_positive_class[0:20])


"""
https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes#5
https://www.kaggle.com/vsundar/amazon-reviews-nb"""
```

```

The top 20 features from the negative class are :

['spread' 'requesting' 'lack resources'
 'certainly control experience school' 'nearly' 'great group' 'journals'
 'teacher_number_of_previously_posted_projects' 'methods' 'love science'
 'mini' 'want students' 'classroom also' 'price' '100 students receive'
 'life ready' 'series' 'properly' 'cooperative learning' 'magazines'] [6083
5645 4353 2664 4949 3919 4298 7061 4849 4639 4864 6863 2758 7062
 2067 4531 5885 5452 2994 4684]
The top 20 features from the positive class are:

['spread' 'requesting' 'lack resources'
 'certainly control experience school' 'nearly' 'journals' 'great group'
 'teacher_number_of_previously_posted_projects' 'love science' 'methods'
 'mini' 'properly' 'want students' 'together' 'price' 'life ready'
 'cooperative learning' '100 students receive' 'classroom also'
 'caucasian'] [6083 5645 4353 2664 4949 4298 3919 7061 4639 4849 4864 5452 6
863 6676
 7062 4531 2994 2067 2758 2645]
```

Out[96]:

```
'\nhttps://stackoverflow.com/questions/50526898/how-to-get-feature-importanc
e-in-naive-bayes#50530697\nhttps://www.kaggle.com/vsundar/amazon-reviews-nb'
```

In [ ]:

```

```

Set 2

In [97]:

```
# Perform Hyperparameter Tuning.
# Plot the training and the CV AUC scores, for different values of 'alpha', using a 2D line

# Please write all the code with proper documentation
from sklearn.feature_extraction.text import TfidfVectorizer

print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
```

```
(22445, 20) (22445,)
(11055, 20) (11055,)
(16500, 20) (16500,)
```

In [98]:

```
# Encoding preprocessed essays
vectorizer_1 = TfidfVectorizer(min_df=10)
vectorizer_1.fit(X_train['preprocessed_essays'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_test_tfidf_essay = vectorizer_1.transform(X_test['preprocessed_essays'].values)
X_train_tfidf_essay = vectorizer_1.transform(X_train['preprocessed_essays'].values)
X_cv_tfidf_essay = vectorizer_1.transform(X_cv['preprocessed_essays'].values)
```

In [99]:

```
print(X_train_tfidf_essay.shape, y_train.shape)
print(X_test_tfidf_essay.shape, y_test.shape)
print(X_cv_tfidf_essay.shape, y_cv.shape)
```

```
(22445, 8782) (22445,)
(16500, 8782) (16500,)
(11055, 8782) (11055,)
```

In [100]:

```
vectorizer_2 = TfidfVectorizer(min_df=10)
vectorizer_2.fit(X_train['preprocessed_title'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_cv_tfidf_title= vectorizer_2.transform(X_cv['preprocessed_title'].values)
X_train_tfidf_title = vectorizer_2.transform(X_train['preprocessed_title'].values)
X_test_tfidf_title = vectorizer_2.transform(X_test['preprocessed_title'].values)
```

In [101]:

```


print(X_train_tfidf_title.shape, y_train.shape)
print(X_cv_tfidf_title.shape, y_cv.shape)
print(X_test_tfidf_title.shape, y_test.shape)
```

```
(22445, 1227) (22445,)
(11055, 1227) (11055,)
(16500, 1227) (16500,)
```

In [102]:

```
# Obtain the optimal value for 'alpha' and using the obtained optimal 'alpha' value, fit a
# Note: If you have split the datase into 3 parts (ie., train, cv and test sets) in the beg
# Make class label and probability predictions on the train and test data.
# preparing Set2
# we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

#**ForSet 2:**
from scipy.sparse import hstack
X_train_set = hstack((X_train_tfidf_title, X_train_tfidf_essay, X_train_categories, X_train
X_test_set = hstack((X_test_tfidf_title, X_test_tfidf_essay, X_test_categories, X_test_scho
X_cv_set= hstack((X_cv_tfidf_title, X_cv_tfidf_essay, X_cv_categories, X_cv_school_state, X
```

In [103]:

```
print(X_train_set.shape, y_train.shape)
print(X_cv_set.shape, y_cv.shape)
print(X_test_set.shape, y_test.shape)
```

```
(22445, 10101) (22445,)
(11055, 10101) (11055,)
(16500, 10101) (16500,)
```

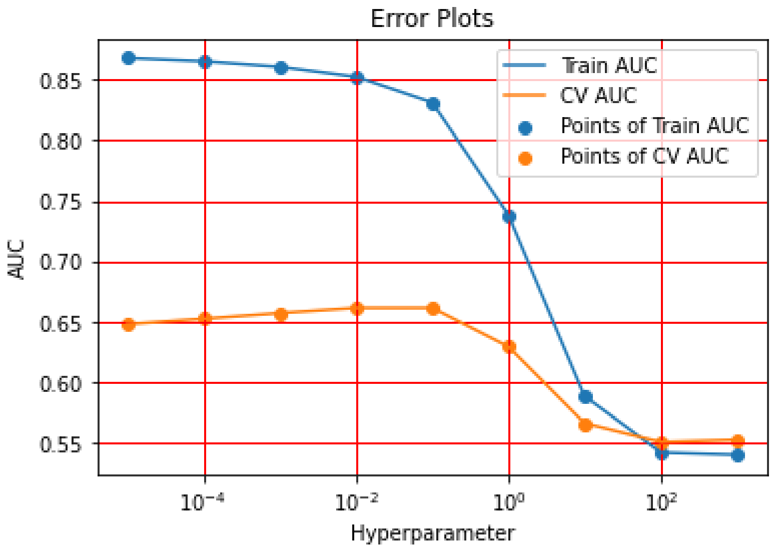
In [104]:

```
# Plot the ROC-AUC curves using the probability predictions made on train and test data.
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

AUC_of_train = []
AUC_of_cv = []
alpha = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10,  100,  1000]
for i in tqdm(alpha):
    clf = MultinomialNB(alpha = i, class_prior = [0.5,0.5])
    clf.fit(X_train_set, y_train)

    y_pred_train = clf.predict_proba(X_train_set)[:,1]
    y_pred_cv = clf.predict_proba(X_cv_set)[:,1]

    # roc_auc_score(y_tr, y_score) the 2nd parameter should be probability estimates of the
    # not the predicted outputs
    AUC_of_train.append(roc_auc_score(y_train,y_pred_train))
    AUC_of_cv.append(roc_auc_score(y_cv, y_pred_cv))
```

```
100%|████████████████████████████████████
██████| 9/9 [00:01<00:00,  6.47it/s]
```

In [105]:

```
plt.semilogx(alpha, AUC_of_train, label='Train AUC')
plt.semilogx(alpha, AUC_of_cv, label='CV AUC')

plt.scatter(alpha, AUC_of_train, label='Points of Train AUC')
plt.scatter(alpha, AUC_of_cv, label='Points of CV AUC')
plt.grid(color='red', linestyle='-', linewidth = 1)
plt.title("Error Plots")
plt.xlabel("Hyperparameter")
plt.ylabel("AUC")
plt.legend()
plt.show()
```



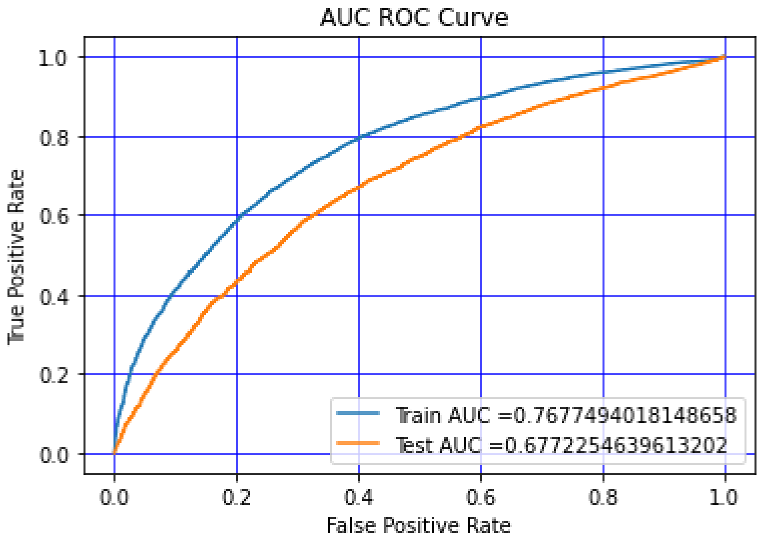
In [106]:

```
alpha_= 100
print("The best Aplha value as :" , alpha_ )
```

```
The best Aplha value as : 100
```

In [107]:

```
plt.plot(FPR_train, TPR_train, label="Train AUC ="+str(auc(FPR_train, TPR_train)))
plt.plot(FPR_test, TPR_test, label="Test AUC ="+str(auc(FPR_test, TPR_test)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC ROC Curve ")
plt.grid(color='blue', linestyle='-', linewidth = 0.7)
plt.show()
```



In [108]:

```
# Pick the best threshold among the probability estimates, such that it has to yield maximu
# Plot the confusion matrices(each for train and test data) afer encoding the predicted cla
def threshoulds_(threshould, FPR, TPR):
    t = threshould[np.argmax(TPR*(1-FPR))]

    print("Maximum value of TPR*(1-FPR)", np.round(max(TPR*(1-FPR)),3), "for threshold", np
    return t
```

In [109]:

```
def prediction_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [110]:

```
from sklearn.metrics import confusion_matrix
best_t = threshoulds_(train_thresholds, FPR_train, TPR_train)

# print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("tn, fp, fn, tp", "=", confusion_matrix(y_train, prediction_with_best_t(y_pred_train,

# print(confusion_matrix(Y_test, predict_with_best_t(y_test_pred, best_t)))
print("tn, fp, fn, tp", "=", confusion_matrix(y_test, prediction_with_best_t(y_pred_test, b
print("here the threshold is ", np.round(best_t,3) , ", i can change the threshold values a
```

```
Maximum value of TPR*(1-FPR) 0.494 for threshold 0.825
tn, fp, fn, tp = [    0  3463     0 18982]
tn, fp, fn, tp = [1476 1070 4351 9603]
here the threshold is  0.825 , i can change the threshold values according t
o Requirement in the confusion matrix
```

In [111]:

```
cm_Train_data = pd.DataFrame(confusion_matrix(y_train, prediction_with_best_t(y_pred_train,
cm_Test_data = pd.DataFrame(confusion_matrix(y_test, prediction_with_best_t(y_pred_test, be
```

In [112]:

```
train_set = hstack((X_train_categories,X_train_teacher_prefix,X_train_school_state,\
        X_train_project_grade_category,X_train_title,X_train_essay_Set1,X_train_norm_price,

cv_set = hstack((X_cv_categories,X_cv_teacher_prefix,X_cv_school_state,X_cv_project_grade_c
                X_cv_norm_price, X_cv_norm, X_cv_essay_Set1, X_cv_title, X_cv_tfidf_essay,

test_set =hstack((X_test_categories,X_test_teacher_prefix,X_test_school_state,\
               X_test_project_grade_category, X_test_essay_Set1, X_test_title,X_test_norm_p
```

In [113]:

```
print(train_set.shape)
print(cv_set.shape)
print(test_set.shape)
```

```
(22445, 17072)
(11055, 18299)
(16500, 17072)
```

In [114]:

```
from sklearn.metrics import roc_curve, auc
neigh = MultinomialNB(alpha = alpha_, class_prior = [0.5,0.5])
neigh.fit(train_set, y_train)
```

Out[114]:

```
MultinomialNB(alpha=100, class_prior=[0.5, 0.5])
```

In [115]:

```
sorted_negative_class_set2 = neigh.feature_log_prob_[0, :].argsort()
sorted_positive_class_set2 = neigh.feature_log_prob_[1, :].argsort()
```

In [116]:

```
from itertools import chain
Stacked_list1 = list(chain(X_train_norm_price, X_train_norm, vectorizer_school_state.get_fe
                           vectorizer_clean_categories.get_feature_names(), vectorizer_proj
                           vectorizer_preprocessed_essays.get_feature_names(), vectorizer_p
```

In [117]:

```
print(np.take(Stacked_list1, sorted_negative_class[-30:-1]))
```

```
[array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.])]
```

In [118]:

```
print(np.take(Stacked_list1, sorted_positive_class[-30:-1]))
```

```
[array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.]) array([1.])
 array([1.]) array([1.]) array([1.]) array([1.]) array([1.])]
```

In [ ]:

3. Summary

as mentioned in the step 5 of instructions

In [119]:

```
#Summarize your assignment work here in a few points, and also compare the final models (fr
# You can either use a pretty table or any other tabular structure.
# Reference Link for Pretty table:  https://pypi.org/project/prettytable/
from prettytable import PrettyTable
```

In [120]:

```
z = PrettyTable()
z.field_names = ["Model","Vectorizer", "Alpha:Hyper Parameter", " Test AUC"]
z.add_row(["Multinomial Naive Bayes", "TFIDF" ,  0.01, 0.68])
z.add_row(["Multinomial Naive Bayes", "BOW", 0.01, 0.77])
```

In [121]:

```
print(z)
```

```
+-------------------------+------------+-----------------------+-----------+
|          Model          | Vectorizer | Alpha:Hyper Parameter |  Test AUC |
+-------------------------+------------+-----------------------+-----------+
| Multinomial Naive Bayes |   TFIDF    |          0.01         |    0.68   |
| Multinomial Naive Bayes |    BOW     |          0.01         |    0.77   |
+-------------------------+------------+-----------------------+-----------+
```

set 2: As AUC is higher , this is giving good idea about the performance of model, AUC of TfIdf Vectorizer is better than BOW.

Higher TPR using thresholds of 0.51, It will create confusion matrix.

Set 1: Higher the TPR with the help of setting the Threshold, Confusion matrix created with thresholds of 0.5.

In [ ]:

In [ ]:

In [ ]:

In [ ]: