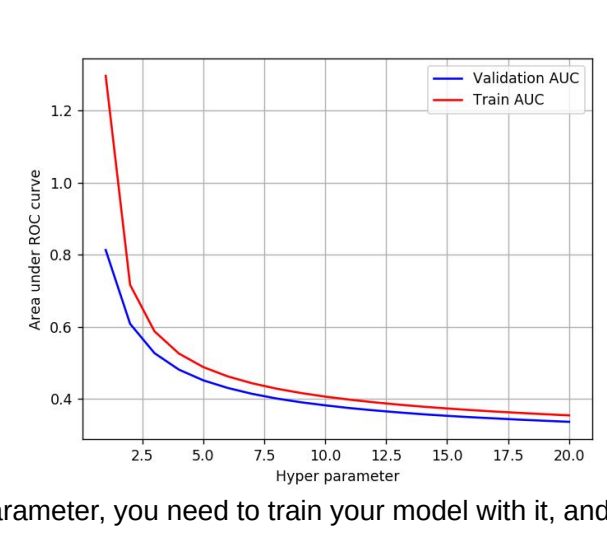
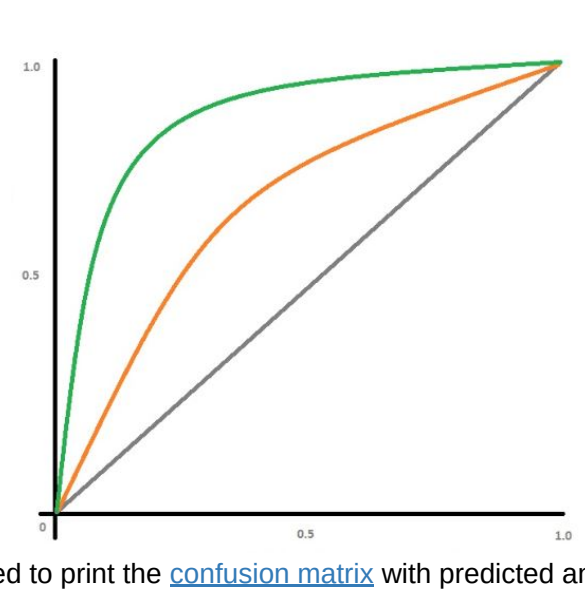
Assignment 6: Apply NB

1. **Apply Multinomial NB on these feature sets**
    - Set 1: categorical, numerical features + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)
2. **The hyper paramter tuning(find best alpha:smoothing parameter)**
    - Find the best hyper parameter which will give the maximum AUC value
    - find the best hyper paramter using k-fold cross validation(use GridsearchCV or RandomsearchCV)/simple cross validation data (write for loop to iterate over hyper parameter values)
    -
3. **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



    - 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 with predicted and original labels of test data points

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

4. fine the top 20 features from either from feature Set 1 or feature Set 2 using absolute 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
5. 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 [101]:
```
%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 chart_studio.plotly import plotly
#from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1. Loading Data

In [102]:
```
project_data = pd.read_csv('train_data.csv',nrows= 100000)
resource_data = pd.read_csv('resources.csv')
```

In [103]:
```
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 (100000, 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 [104]:
```
# 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.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
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[104]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_sub
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	

In [105]:
```
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[105]:

	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. Preprocessing

Preprocessing of `project_subject_categories`

In [106]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# 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
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", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & 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:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

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

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

Preprocessing of `project_subject_subcategories`

In [107]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# 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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & 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:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        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]))
```

Text preprocessing

In [108]:
```
# 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 [109]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom as well as the STEM journals, which my students really enjoyed.  I would love to implement more of the Lakeshore STEM kits in my classroom for the next school year as they provide excellent and engaging STEM lessons.My students come from a variety of backgrounds, including language and socioeconomic status.  Many of them don't have a lot of experience in science and engineering and these kits give me the materials to provide these exciting opportunities for my students.Each month I try to do several science or STEM/STEAM projects.  I would use the kits and robot to help guide my science instruction in engaging and meaningful ways.  I can adapt the kits to my current language arts pacing guide where we already teach some of the material in the kits like tall tales (Paul Bunyan) or Johnny Appleseed.  The following units will be taught in the next school year where I will implement these kits: magnets, motion, sink vs. float, robots.  I often get to these units and don't know If I am teaching the right way or using the right materials.    The kits will give me additional ideas, strategies, and lessons to prepare my students in science.It is challenging to develop high quality science activities.  These kits give me the materials I need to provide my students with science activities that will go along with the curriculum in my classroom.  Although I have some things (like magnets) in my classroom, I don't know how to use them effectively.  The kits will provide me with the right amount of materials and show me how to use them in an appropriate way.
==================================================
My class enters the room on a daily basis VERY eager to learn. We are an active group of students who deserve every opportunity as every other school district in our state. Technology is our FUTURE!!! The children in the city  in which I teach will GREATLY benefit from the use of this  technology.We are located in Camden, New Jersey which has been described by the media as \"the poorest city in the US\" and \"the most dangerous city in America.\" A full 96% of our students are classified as \"economically disadvantaged,\" 16% have special needs. My 17 students in my class are Awesome!!!!  I am a 6th grade teacher in an urban school district. My school is a fabulous school in which I believe is the BEST in the city. We are a close knit school, in a close knit community. Parents, teachers, and community member TRULY care about the well-being of our children.My students come to school every day, and are EXTREMELY eager to learn new things.My students are over achievers, who deserve the best education that they can!!!! It is my job, as their teacher to ensure that I prepare my students for their future. By integrating technology into our classroom will only enhance their love of learning.Presently, I have 4 computers in my classroom. With 17 students, it is very difficult to accommodate all of the students during research projects, etc.....  I believe by adding CHROMES to our classroom will only enhance the students love of learning in our classroom. Our students will be able to engage in all educational learning activities including research, educational experiences such as EDMEMTON, as well as allowing all  students to experience writing websites, as well as prepare them for the PARCC. The students will be able to engage themselves with their Chromes, and they will not have to \"wait their turn,\" for a computer. By adding 4 more means of technology into our classroom will only benefit our AWESOME class and turn them into \"higher level learners, and thinkers.\" My students deserve every opportunity possible. I am preparing them for their future,  and the future  is TECHNOLOGY!!!! This project will take my student's love of learning to the next level,This project will make a difference because I believe the AWESOME students in my class SOOOOO DESERVE THE OPPORTUNITY TO INTEGRATE TECHNOLOGY INTO OUR CLASSROOM.  With 4 computers, it is extremely difficult for all students to engage in activities in which entail technology The FUTURE is technology, and WE DESERVE to be \"up to date,\" with the times!!
==================================================
Remember working hard to earn a certificate or degree?  Or maybe it was just an honor roll certificate that put a smile on your face.  Every little achievement my students make, we celebrate! They work hard to successfully meet standards and to go out and present themselves in a professional manner.Our school is majority free and reduced price lunch. The district is constantly experiencing drastic cuts. My students walk into class ready to learn every morning! No matter what gets them down, their curiosity about the day's activities brighten their mood. They thrive on meeting the challenges they are faced with in their daily projects and activities. If they only had more supplies to allow them more freedom in exploring their creativity on their way to becoming inspiring teachers they'd face an even biggerStudents will use these materials to practice the skills and scenarios necessary in obtaining their CPR certification. The ideal ratio for students to manikins is 1:1.  With these few, we would have a ratio of 2:1 which will assist us meeting our needs.Donations to this project will allow students to be trained in CPR and first aid which will allow them to go out into our community and be of assistance if and when a situation may occur.
==================================================
My students come from hardworking, loving families.  I have students from all socioeconomic classes and many ethnic backgrounds.  The families of our students are loving and involved in their kids' lives.  Unfortunately several of our families work more than one job to make ends meet and some come from single parent homes.  With this being said our parents still make time to work with their kids on homework and nightly reading.  They know how important school is for their children.  My students love to learn and are ready the minute they walk through the door.  I want to make the absolute most of my time with them.\r\n\r\nAt this age kids have their own ideas of what school should be and if we listen to them they will inspire us to make school more fun!!!\r\n\r\nMy students learn best through play.  They enjoy games and activities that will teach them about real life events and get them up and moving.  Anytime they are not sitting at their desk they are so much happier and want to learn so much more. They think they are just having fun  but teachers know they are learning, Win Win!In Tennessee we have all kinds of crazy weather.  We need to be prepared for indoor recess all year long.  My students are always sharing ideas with me about what they have had at other schools or aftercare programs and would love some of those items in our classroom to make indoor recess fun for everyone.  They do not want to sit still and watch a video .  They are wanting to roam from Fun activity to another Fun activity.  If these are things they want I can even put an educational spin on it and they will never know.  \r\n\r\nThey want light boxes and building toys and tiles that are magnetic to build and explore all during recess, but they also want a tablet to be able to sing and dance with Jack Hartman and Dr. Jean anytime !!  These items will keep them up and busy during a time when they can not be outside due to weather.  We can also use them to make our Literacy ,Math and Science centers more active.  The hokki stools are a special request from them since other teachers have them and they can get their wiggles our while still doing their work.  All of these items are able to be used throughout Kindergarten to keep our kids excited about learning and getting them moving and working together to build and make things they are interested in. Our students will be so much happier when they hear \"indoor recess today\" or \" it's raining again\" and they will still get their exercise and play time in!!\r\n\r\n\r\nnannan
==================================================
My first graders are eager to learn about the world around them.  They come to school each day full of enthusiasm and genuinely love learning. \r\n\r\nOur diverse class includes students from a variety of cultural and economic backgrounds.  Many come from homes where parents can't afford or simply don't know the importance of books, so it is important to me to provide an environment that is rich in literature so that students learn to love reading.  I want my students to be lifelong learners, and reading is the best way!I have used these magazines in the past, and kids absolutely LOVE them!! The topics are of high interest for children and always correspond to real world issues that are important for kids to learn. The subscription also includes online resources such as videos, printable worksheets, and skill-based games. \r\n\r\nThese materials will expose students to rigorous and interesting nonfiction text that will spark their curiosity about the world around them.  The topics allow me to teach the nonfiction text standards using interesting materials.  They always lead to engaging discussions and inspire students to find additional information about the various topics.nannan
==================================================
```

In [110]:
```
# 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 [111]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My students come from hardworking, loving families.  I have students from all socioeconomic classes and many ethnic backgrounds.  The families of our students are loving and involved in their kids' lives.  Unfortunately several of our families work more than one job to make ends meet and some come from single parent homes.  With this being said our parents still make time to work with their kids on homework and nightly reading.  They know how important school is for their children.  My students love to learn and are ready the minute they walk through the door.  I want to make the absolute most of my time with them.\r\n\r\nAt this age kids have their own ideas of what school should be and if we listen to them they will inspire us to make school more fun!!!\r\n\r\nMy students learn best through play.  They enjoy games and activities that will teach them about real life events and get them up and moving.  Anytime they are not sitting at their desk they are so much happier and want to learn so much more. They think they are just having fun  but teachers know they are learning, Win Win!In Tennessee we have all kinds of crazy weather.  We need to be prepared for indoor recess all year long.  My students are always sharing ideas with me about what they have had at other schools or aftercare programs and would love some of those items in our classroom to make indoor recess fun for everyone.  They do not want to sit still and watch a video .  They are wanting to roam from Fun activity to another Fun activity.  If these are things they want I can even put an educational spin on it and they will never know.  \r\n\r\nThey want light boxes and building toys and tiles that are magnetic to build and explore all during recess, but they also want a tablet to be able to sing and dance with Jack Hartman and Dr. Jean anytime !!  These items will keep them up and busy during a time when they can not be outside due to weather.  We can also use them to make our Literacy ,Math and Science centers more active.  The hokki stools are a special request from them since other teachers have them and they can get their wiggles our while still doing their work.  All of these items are able to be used throughout Kindergarten to keep our kids excited about learning and getting them moving and working together to build and make things they are interested in. Our students will be so much happier when they hear \"indoor recess today\" or \" it is raining again\" and they will still get their exercise and play time in!!\r\n\r\n\r\nnannan
==================================================
```

In [112]:
```
# \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 students come from hardworking, loving families.  I have students from all socioeconomic classes and many ethnic backgrounds.  The families of our students are loving and involved in their kids' lives.  Unfortunately several of our families work more than one job to make ends meet and some come from single parent homes.  With this being said our parents still make time to work with their kids on homework and nightly reading.  They know how important school is for their children.  My students love to learn and are ready the minute they walk through the door.  I want to make the absolute most of my time with them.    At this age kids have their own ideas of what school should be and if we listen to them they will inspire us to make school more fun!!!    My students learn best through play.  They enjoy games and activities that will teach them about real life events and get them up and moving.  Anytime they are not sitting at their desk they are so much happier and want to learn so much more. They think they are just having fun  but teachers know they are learning, Win Win!In Tennessee we have all kinds of crazy weather.  We need to be prepared for indoor recess all year long.  My students are always sharing ideas with me about what they have had at other schools or aftercare programs and would love some of those items in our classroom to make indoor recess fun for everyone.  They do not want to sit still and watch a video .  They are wanting to roam from Fun activity to another Fun activity.  If these are things they want I can even put an educational spin on it and they will never know.    They want light boxes and building toys and tiles that are magnetic to build and explore all during recess, but they also want a tablet to be able to sing and dance with Jack Hartman and Dr. Jean anytime !!  These items will keep them up and busy during a time when they can not be outside due to weather.  We can also use them to make our Literacy ,Math and Science centers more active.  The hokki stools are a special request from them since other teachers have them and they can get their wiggles our while still doing their work.  All of these items are able to be used throughout Kindergarten to keep our kids excited about learning and getting them moving and working together to build and make things they are interested in. Our students will be so much happier when they hear  indoor recess today  or  it is raining again  and they will still get their exercise and play time in!!      nannan
```

In [113]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My students come from hardworking loving families I have students from all socioeconomic classes and many ethnic backgrounds The families of our students are loving and involved in their kids lives Unfortunately several of our families work more than one job to make ends meet and some come from single parent homes With this being said our parents still make time to work with their kids on homework and nightly reading They know how important school is for their children My students love to learn and are ready the minute they walk through the door I want to make the absolute most of my time with them At this age kids have their own ideas of what school should be and if we listen to them they will inspire us to make school more fun My students learn best through play They enjoy games and activities that will teach them about real life events and get them up and moving Anytime they are not sitting at their desk they are so much happier and want to learn so much more They think they are just having fun but teachers know they are learning Win Win In Tennessee we have all kinds of crazy weather We need to be prepared for indoor recess all year long My students are always sharing ideas with me about what they have had at other schools or aftercare programs and would love some of those items in our classroom to make indoor recess fun for everyone They do not want to sit still and watch a video They are wanting to roam from Fun activity to another Fun activity If these are things they want I can even put an educational spin on it and they will never know They want light boxes and building toys and tiles that are magnetic to build and explore all during recess but they also want a tablet to be able to sing and dance with Jack Hartman and Dr Jean anytime These items will keep them up and busy during a time when they can not be outside due to weather We can also use them to make our Literacy Math and Science centers more active The hokki stools are a special request from them since other teachers have them and they can get their wiggles our while still doing their work All of these items are able to be used throughout Kindergarten to keep our kids excited about learning and getting them moving and working together to build and make things they are interested in Our students will be so much happier when they hear indoor recess today or it is raining again and they will still get their exercise and play time in nannan
```

In [114]:
```
# 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 [115]:
```
# 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%|██████████| 100000/100000 [01:09<00:00, 1441.93it/s]
```

In [116]:
```
print(project_data['essay'].values[5000])
```

```
Teaching fourth graders is definitely a challenging task each day but one I wake up ready to tackle. My fourth graders are bright, kind, and amazingly sweet kiddos. They are creative, innovative, and willing to tackle any task I throw their way. They are technologically savvy, and love to move! \r\n\r\nMy students love choices and love to move to do their work. They love to read in different spots around the classroom, and use clipboards. The best part of my job is watching them grow and learn each day! \r\n\r\nWe always hear that we should give our kiddos choices. Yet, we rarely give them choices of when and how they sit in class all day. My students sit in a hard chair or on the hard linoleum floor. Yet, neither option is comfortable. \r\n\r\nMy goal is to help my students be more comfortable by giving them a variety of choices. I want to give them options to sit on a bean bag chair or use a lap desk on a carpet square, or sit on a bar stool at a standing desk. Kids crave options and choices just like we do. They need those options to learn more and grow each day. By giving them just a few different options, my students will grow more and be able to actually be comfortable in their classroom.nannan
```

In [117]:
```
# after preprocesing
preprocessed_essays[5000]
```

Out[117]:
```
'teaching fourth graders definitely challenging task day one wake ready tackle fourth graders bright kind amazingly sweet kiddos creative innovative willing tackle task throw way technologically savvy love move students love choices love move work love read different spots around classroom use clipboards best part job watching grow learn day always hear give kiddos choices yet rarely give choices sit class day students sit hard chair hard linoleum floor yet neither option comfortable goal help students comfortable giving variety choices want give options sit bean bag chair use lap desk carpet square sit bar stool standing desk kids crave options choices like need options learn grow day giving different options students grow able actually comfortable classroom nannan'
```

In [118]:
```
# after preprocesing
project_data['preprocessed_essays'] = preprocessed_essays
```

In [119]:
```
print(project_data['preprocessed_essays'].values[5000])
```

```
teaching fourth graders definitely challenging task day one wake ready tackle fourth graders bright kind amazingly sweet kiddos creative innovative willing tackle task throw way technologically savvy love move students love choices love move work love read different spots around classroom use clipboards best part job watching grow learn day always hear give kiddos choices yet rarely give choices sit class day students sit hard chair hard linoleum floor yet neither option comfortable goal help students comfortable giving variety choices want give options sit bean bag chair use lap desk carpet square sit bar stool standing desk kids crave options choices like need options learn grow day giving different options students grow able actually comfortable classroom nannan
```

Preprocessing of `project_title`

In [120]:
```
# similarly you can preprocess the titles also

# printing some random titles.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
print(project_data['project_title'].values[200])
print("="*50)
print(project_data['project_title'].values[500])
print("="*50)
```

```
Engineering STEAM into the Primary Classroom
==================================================
\"Bringing the FUTURE into our Classroom\"
==================================================
The Outdoor Classroom
==================================================
Mini iPads for Many Fingers
==================================================
ILT loves PE!!
==================================================
```

In [121]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_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_title.append(sent.lower().strip())
```

```
100%|██████████| 100000/100000 [00:03<00:00, 30765.15it/s]
```

In [122]:
```
# after preprocesing
preprocessed_title[100]
```

Out[122]:
```
'the outdoor classroom'
```

In [123]:
```
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
print(project_data['project_title'].values[200])
print("="*50)
print(project_data['project_title'].values[500])
print("="*50)
```

```
Engineering STEAM into the Primary Classroom
==================================================
\"Bringing the FUTURE into our Classroom\"
==================================================
The Outdoor Classroom
==================================================
Mini iPads for Many Fingers
==================================================
ILT loves PE!!
==================================================
```

In [124]:
```
project_data['preprocessed_title'] = preprocessed_title
```

In [125]:
```
print(project_data['preprocessed_title'].values[0])
print("="*50)
print(project_data['preprocessed_title'].values[150])
print("="*50)
print(project_data['preprocessed_title'].values[100])
print("="*50)
print(project_data['preprocessed_title'].values[200])
print("="*50)
print(project_data['preprocessed_title'].values[500])
print("="*50)
```

```
engineering steam primary classroom
==================================================
bringing future classroom
==================================================
the outdoor classroom
==================================================
mini ipads many fingers
==================================================
ilt loves pe
==================================================
```

Preprocessing categorical features : project_grade_category

In [126]:
```
project_data['project_grade_category'].value_counts()
```

Out[126]:
```
Grades PreK-2    40460
Grades 3-5       34028
Grades 6-8       15478
Grades 9-12      10034
Name: project_grade_category, dtype: int64
```

In [127]:
```
# Mention code reference here
# we need to remove spaces, replace '-' with '_' and conver all letters to small
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[127]:
```
grades_prek_2    40460
grades_3_5       34028
grades_6_8       15478
grades_9_12      10034
Name: project_grade_category, dtype: int64
```

Preprocessing categorical features : teacher_prefix

In [128]:
```
project_data['teacher_prefix'].value_counts()
```

Out[128]:
```
Mrs.       52403
Ms.        35690
Mr.         9733
Teacher     2158
Dr.           13
Name: teacher_prefix, dtype: int64
```

In [129]:
```
print(project_data['teacher_prefix'].isnull().values.any())
print("Number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
True
Number of nan values 3
```

In [130]:
```
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('Mrs.')
```

In [131]:
```
project_data['teacher_prefix'].value_counts()
```

Out[131]:
```
Mrs.       52406
Ms.        35690
Mr.         9733
Teacher     2158
Dr.           13
Name: teacher_prefix, dtype: int64
```

In [132]:
```
#remove '.' and convert chars to small
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()
```

In [133]:
```
project_data['teacher_prefix'].value_counts()
```

Out[133]:
```
mrs        52406
ms         35690
mr          9733
teacher     2158
dr            13
Name: teacher_prefix, dtype: int64
```

Preprocessing categorical features : school_state

In [134]:
```
project_data['school_state'].value_counts()
```

Out[134]:
```
CA    14090
TX     6764
NY     6704
FL     5693
NC     4651
IL     3983
GA     3630
SC     3594
MI     2886
PA     2847
IN     2409
MO     2368
OH     2273
MA     2183
LA     2164
WA     2130
OK     2111
NJ     2042
AZ     1969
VA     1872
WI     1668
AL     1606
UT     1584
TN     1545
CT     1511
MD     1384
NV     1252
MS     1198
KY     1197
OR     1141
MN     1100
CO     1024
AR      946
ID      633
IA      606
KS      581
NM      498
DC      477
WV      472
HI      468
ME      464
AK      321
DE      313
NH      312
NE      288
SD      269
RI      264
MT      216
ND      134
WY       90
VT       75
Name: school_state, dtype: int64
```

In [135]:
```
#convert school state to lower case
project_data['school_state'] = project_data['school_state'].str.lower()
project_data['school_state'].value_counts()
```
Out[135]:
```
ca    14090
tx     6764
ny     6704
fl     5693
nc     4651
il     3983
ga     3630
sc     3594
mi     2886
pa     2847
in     2409
mo     2368
oh     2273
ma     2183
la     2164
wa     2130
ok     2111
nj     2042
az     1969
va     1872
wi     1668
al     1606
ut     1584
tn     1545
ct     1511
md     1384
nv     1252
ms     1198
ky     1197
or     1141
mn     1100
co     1024
ar      946
id      633
ia      606
ks      581
nm      498
dc      477
wv      472
hi      468
me      464
ak      321
de      313
nh      312
ne      288
sd      269
ri      264
mt      216
nd      134
wy       90
vt       75
Name: school_state, dtype: int64
```

In [136]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```
Out[136]:

	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 [137]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```
Out[137]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [138]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [139]:
```
project_data.columns
```
Out[139]:
```
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_approved',
       'clean_categories', 'clean_subcategories', 'essay',
       'preprocessed_essays', 'preprocessed_title', 'price', 'quantity'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

In [140]:
```
project_data.head(2)
```
Out[140]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_title	pr
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	mrs	ca	2016-04-27 00:27:36	grades_prek_2	Engineering STEAM into the Primary Classroom	fo
1	37728	p043609	3f60494c61921b3b43ab61bdde2904df	ms	ut	2016-04-27 00:31:25	grades_3_5	Sensory Tools for Focus	In

2 rows × 22 columns

2. Naive Bayes

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

In [145]:
```
y = project_data['project_is_approved'].values
X = project_data.drop(['project_is_approved'], axis=1)
X.head(2)

# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)


print("="*100)
```
```
====================================================================================================
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [146]:
```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```
```
(44890, 21) (44890,)
(22110, 21) (22110,)
(33000, 21) (33000,)
====================================================================================================
```

Vectorizing Categorical features

Vectorizing : clean_categories

In [147]:
```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_clean_categories = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_clean_categories.fit(X_train['clean_categories'].values)
print(vectorizer_clean_categories.get_feature_names())


X_train_categories_one_hot = vectorizer_clean_categories.transform(X_train['clean_categories'].values)
X_cv_categories_one_hot = vectorizer_clean_categories.transform(X_cv['clean_categories'].values)
X_test_categories_one_hot = vectorizer_clean_categories.transform(X_test['clean_categories'].values)


print("Shape of matrix after one hot encodig ")
print(X_train_categories_one_hot.shape, y_train.shape)
print(X_cv_categories_one_hot.shape, y_cv.shape)
print(X_test_categories_one_hot.shape, y_test.shape)
```
```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig
(44890, 9) (44890,)
(22110, 9) (22110,)
(33000, 9) (33000,)
```

Vectorizing : clean_subcategories

In [148]:
```
# Vectorizing clean_subcategories
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_clean_subcategories = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_clean_subcategories.fit(X_train['clean_subcategories'].values)

X_train_sub_categories_one_hot = vectorizer_clean_subcategories.transform(X_train['clean_subcategories'].values)
X_cv_sub_categories_one_hot = vectorizer_clean_subcategories.transform(X_cv['clean_subcategories'].values)
X_test_sub_categories_one_hot = vectorizer_clean_subcategories.transform(X_test['clean_subcategories'].values)

print("Shape of matrix after one hot encodig ")
print(X_train_sub_categories_one_hot.shape, y_train.shape)
print(X_cv_sub_categories_one_hot.shape, y_cv.shape)
print(X_test_sub_categories_one_hot.shape, y_test.shape)
```
```
Shape of matrix after one hot encodig
(44890, 30) (44890,)
(22110, 30) (22110,)
(33000, 30) (33000,)
```

Vectorizing : school_state

In [149]:
```
# 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)

X_train_school_state_one_hot = vectorizer_school_state.transform(X_train['school_state'].values)
X_cv_school_state_one_hot = vectorizer_school_state.transform(X_cv['school_state'].values)
X_test_school_state_one_hot = vectorizer_school_state.transform(X_test['school_state'].values)

print("Shape of matrix after one hot encodig ")
print(X_train_school_state_one_hot.shape, y_train.shape)
print(X_cv_school_state_one_hot.shape, y_cv.shape)
print(X_test_school_state_one_hot.shape, y_test.shape)
print("="*100)
```
```
Shape of matrix after one hot encodig
(44890, 51) (44890,)
(22110, 51) (22110,)
(33000, 51) (33000,)
====================================================================================================
```

Vectorizing : teacher_prefix

In [150]:
```
#teacher_prefix encoding
vectorizer_teacher_prefix = CountVectorizer(binary=True)
vectorizer_teacher_prefix.fit(X_train['teacher_prefix'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_prefix_one_hot = vectorizer_teacher_prefix.transform(X_train['teacher_prefix'].values)
X_cv_teacher_prefix_one_hot = vectorizer_teacher_prefix.transform(X_cv['teacher_prefix'].values)
X_test_teacher_prefix_one_hot = vectorizer_teacher_prefix.transform(X_test['teacher_prefix'].values)

print("Shape of matrix after one hot encodig ")
print(X_train_teacher_prefix_one_hot.shape, y_train.shape)
print(X_cv_teacher_prefix_one_hot.shape, y_cv.shape)
print(X_test_teacher_prefix_one_hot.shape, y_test.shape)

print("="*100)
```
```
Shape of matrix after one hot encodig
(44890, 5) (44890,)
(22110, 5) (22110,)
(33000, 5) (33000,)
====================================================================================================
```

Vectorizing : project_grade_category

In [151]:
```
#encoding project_grade_category
vectorizer_project_grade_category = CountVectorizer(binary=True)
vectorizer_project_grade_category.fit(X_train['project_grade_category'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_project_grade_category_one_hot= vectorizer_project_grade_category.transform(X_train['project_grade_category'].values)
X_cv_project_grade_category_one_hot= vectorizer_project_grade_category.transform(X_cv['project_grade_category'].values)
X_test_project_grade_category_one_hot= vectorizer_project_grade_category.transform(X_test['project_grade_category'].values)


print("Shape of matrix after one hot encodig ")
print(X_train_project_grade_category_one_hot.shape, y_train.shape)
print(X_cv_project_grade_category_one_hot.shape, y_cv.shape)
print(X_test_project_grade_category_one_hot.shape, y_test.shape)

print("="*100)
```
```
Shape of matrix after one hot encodig
(44890, 4) (44890,)
(22110, 4) (22110,)
(33000, 4) (33000,)
====================================================================================================
```

Encoding numerical features

Encoding numerical feature: Price

In [152]:
```
# Encoding Price feature
# 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))
X_cv_price_standardized = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
X_test_price_standardized = price_scalar.transform(X_test['price'].values.reshape(-1, 1))

X_train_price_standardized
X_cv_price_standardized
X_test_price_standardized


print("Shape of matrix after encodig ")
print(X_train_price_standardized.shape, y_train.shape)
print(X_cv_price_standardized.shape, y_cv.shape)
print(X_test_price_standardized.shape, y_test.shape)

"""

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(-1,1))

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

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```
```
After vectorizations
(44890, 1) (44890,)
(22110, 1) (22110,)
(33000, 1) (33000,)
====================================================================================================
```

Vectorizing : teacher_number_of_previously_posted_projects : numerical

In [154]:
```
# Encoding teacher_number_of_previously_posted_projects : numerical feature
# 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)

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

# Now standardize the data with above maen and variance.
X_train_teacher_number_of_previously_posted_projects_standardized = teacher_number_of_previously_posted_projects_scalar.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
X_cv_teacher_number_of_previously_posted_projects_standardized = teacher_number_of_previously_posted_projects_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
X_test_teacher_number_of_previously_posted_projects_standardized = teacher_number_of_previously_posted_projects_scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))

X_train_teacher_number_of_previously_posted_projects_standardized
X_cv_teacher_number_of_previously_posted_projects_standardized
X_test_teacher_number_of_previously_posted_projects_standardized

print("Shape of the matrix after vectorization:")
print(X_train_teacher_number_of_previously_posted_projects_standardized.shape,y_train.shape)
print(X_cv_teacher_number_of_previously_posted_projects_standardized.shape,y_cv.shape)
print(X_test_teacher_number_of_previously_posted_projects_standardized.shape,y_test.shape)"""

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_teacher_number_of_previously_posted_projects_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_cv_teacher_number_of_previously_posted_projects_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_teacher_number_of_previously_posted_projects_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_teacher_number_of_previously_posted_projects_norm.shape, y_train.shape)
print(X_cv_teacher_number_of_previously_posted_projects_norm.shape, y_cv.shape)
print(X_test_teacher_number_of_previously_posted_projects_norm.shape, y_test.shape)
print("="*100)
```
```
After vectorizations
(44890, 1) (44890,)
(22110, 1) (22110,)
(33000, 1) (33000,)
====================================================================================================
```

Vectozing Text data using BOW

Encoding : preprocessed_essays

In [155]:
```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

#Encoding preprocessed essays using BOW

vectorizer_preprocessed_essays = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer_preprocessed_essays.fit(X_train['preprocessed_essays'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer_preprocessed_essays.transform(X_train['preprocessed_essays'].values)
X_cv_essay_bow = vectorizer_preprocessed_essays.transform(X_cv['preprocessed_essays'].values)
X_test_essay_bow = vectorizer_preprocessed_essays.transform(X_test['preprocessed_essays'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```
```
(44890, 21) (44890,)
(22110, 21) (22110,)
(33000, 21) (33000,)
====================================================================================================
After vectorizations
(44890, 5000) (44890,)
(22110, 5000) (22110,)
(33000, 5000) (33000,)
====================================================================================================
```

Encoding : preprocessed_title

In [157]:
```
# encoding preprocessed title using BOW
print("="*100)


vectorizer_preprocessed_title = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer_preprocessed_title.fit(X_train['preprocessed_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_bow = vectorizer_preprocessed_title.transform(X_train['preprocessed_title'].values)
X_cv_title_bow = vectorizer_preprocessed_title.transform(X_cv['preprocessed_title'].values)
X_test_title_bow = vectorizer_preprocessed_title.transform(X_test['preprocessed_title'].values)

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
print("="*100)
```
```
====================================================================================================
After vectorizations
(44890, 3779) (44890,)
(22110, 3779) (22110,)
(33000, 3779) (33000,)
====================================================================================================
```

2.3 Merging all the above features

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

In [158]:
```
#we need to merge all the numerical vectors i.e catogorical, text, numerical vectors
from scipy.sparse import hstack
X_train_set1 = hstack((X_train_title_bow, X_train_essay_bow, X_train_categories_one_hot, X_train_sub_categories_one_hot, X_train_school_state_one_hot, X_train_teacher_prefix_one_hot, X_train_project_grade_category_one_hot, X_train_price_norm,X_train_teacher_number_of_previously_posted_projects_norm)).tocsr()
X_cv_set1= hstack((X_cv_title_bow, X_cv_essay_bow, X_cv_categories_one_hot, X_cv_sub_categories_one_hot, X_cv_school_state_one_hot, X_cv_teacher_prefix_one_hot, X_cv_project_grade_category_one_hot, X_cv_price_norm, X_cv_teacher_number_of_previously_posted_projects_norm)).tocsr()
X_test_set1 = hstack((X_test_title_bow, X_test_essay_bow, X_test_categories_one_hot, X_test_sub_categories_one_hot, X_test_school_state_one_hot, X_test_teacher_prefix_one_hot, X_test_project_grade_category_one_hot, X_test_price_norm, X_test_teacher_number_of_previously_posted_projects_norm)).tocsr()

print("Final Data matrix")
print(X_train_set1.shape, y_train.shape)
print(X_cv_set1.shape, y_cv.shape)
print(X_test_set1.shape, y_test.shape)
print("="*100)
```
```
Final Data matrix
(44890, 8880) (44890,)
(22110, 8880) (22110,)
(33000, 8880) (33000,)
====================================================================================================
```

2.4 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 instrucations

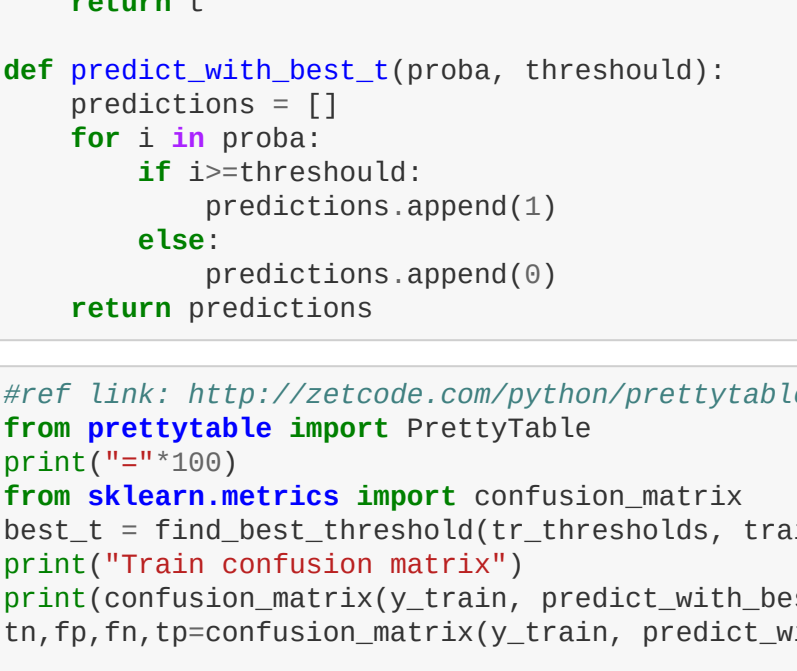
2.4.1 Applying NB on BOW featurization, SET 1

Method 1: Simple for loop (if you are having memory limitations use this)

In [159]:
```
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
```

Hyperparameter tuning

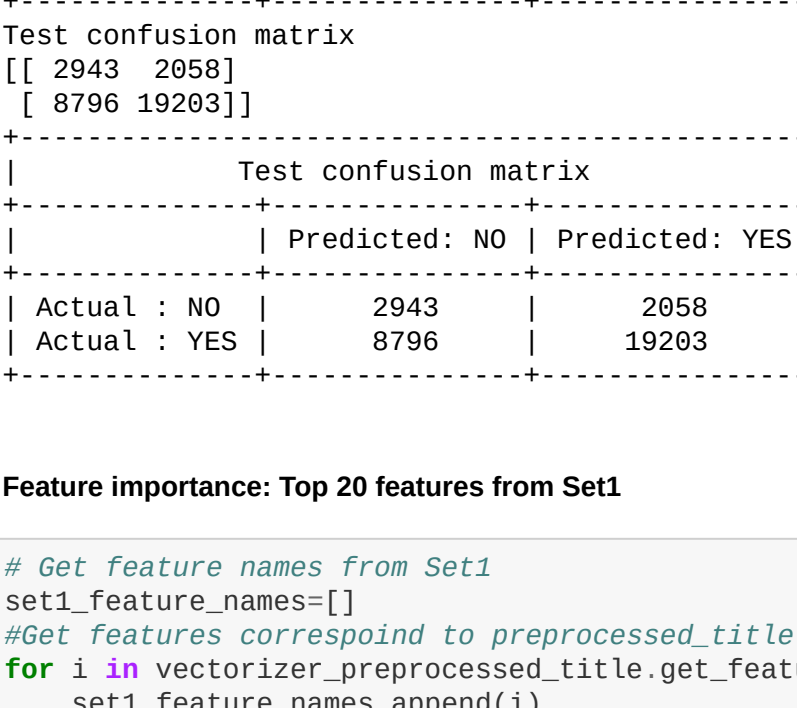
In [160]:
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thresholded measure of
decisions (as returned by “decision_function” on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
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_train_pred = batch_predict(model, X_train_set1)
    y_cv_pred = batch_predict(model, X_cv_set1)

    # 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))

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

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

plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```
```
100%|██████████| 6/6 [00:01<00:00,  3.58it/s]
```



Testing the performance of the model on test data, plotting ROC Curves

In [161]:
```
#here we are choosing best value of alpha based on for loop results
best_alpha = 0.1
```

In [162]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


model = MultinomialNB(alpha=best_alpha)
model.fit(X_train_set1, 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(model, X_train_set1)
y_test_pred = batch_predict(model, X_test_set1)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FALSE POSITIVE RATE(FPR)")
plt.ylabel("TRUE POSITIVE RATE(TPR)")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



Confusion Matrix

In [163]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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,3))
    return t

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

In [164]:
```
#ref link: http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
tn,fp,fn,tp=confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)).ravel()

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

print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
tn,fp,fn,tp=confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)).ravel()
tb = PrettyTable()
tb.title= 'Test confusion matrix'
tb.field_names= ("","Predicted: NO" ,"Predicted: YES")
tb.add_row(["Actual : NO",tn,fp ])
tb.add_row(["Actual : YES", fn,tp])
print(tb)
```
```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.47107290637474236 for threshold 0.78
Train confusion matrix
[[ 4603  2200]
 [11570 26517]]
+----------------------------------------------+
|            Train confusion matrix            |
+--------------+---------------+----------------+
|              | Predicted: NO | Predicted: YES |
+--------------+---------------+----------------+
| Actual : NO  |      4603     |      2200      |
| Actual : YES |     11570     |     26517      |
+--------------+---------------+----------------+
Test confusion matrix
[[ 2943  2058]
 [ 8796 19203]]
+----------------------------------------------+
|            Test confusion matrix             |
+--------------+---------------+----------------+
|              | Predicted: NO | Predicted: YES |
+--------------+---------------+----------------+
| Actual : NO  |      2943     |      2058      |
| Actual : YES |      8796     |     19203      |
+--------------+---------------+----------------+
```

Feature importance: Top 20 features from Set1

In [165]:
```
# Get feature names from Set1
set1_feature_names=[]
#Get features correspoind to preprocessed_title
for i in vectorizer_preprocessed_title.get_feature_names():
    set1_feature_names.append(i)
#Get features correspoind to preprocessed_essays
for i in vectorizer_preprocessed_essays.get_feature_names():
    set1_feature_names.append(i)
#Get features correspoind to clean_categories
for i in vectorizer_clean_categories.get_feature_names():
    set1_feature_names.append(i)
#Get features correspoinding to clean_subcategories
for i in vectorizer_clean_subcategories.get_feature_names():
    set1_feature_names.append(i)
#Get features correspoinding to school_state
for i in vectorizer_school_state.get_feature_names():
    set1_feature_names.append(i)
#Get features correspoinding to teacher_prefix
for i in vectorizer_teacher_prefix.get_feature_names():
    set1_feature_names.append(i)
#Get features correspoinding to project_grade_category
for i in vectorizer_project_grade_category.get_feature_names():
    set1_feature_names.append(i)
set1_feature_names.append("price")
set1_feature_names.append("teacher_number_of_previously_posted_projects")

#print(set1_feature_names)
print(len(set1_feature_names))
```
```
8880
```

In [166]:
```
# Get top 20 features displayed for both the classes
"""Code Ref:https:
https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes#50530697
https://www.kaggle.com/vsundar/amazon-reviews-nb"""

neg_class_prob_sorted = (-model.feature_log_prob_[0, :]).argsort()
pos_class_prob_sorted = (-model.feature_log_prob_[1, :]).argsort()

neg_class_features = np.take(set1_feature_names, neg_class_prob_sorted[:20])
pos_class_features = np.take(set1_feature_names, pos_class_prob_sorted[:20])


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

print("\nThe top 20 features from the negative class are :\n")
print(neg_class_features, neg_class_prob_sorted[:20])
```
```
The top 20 features from the positive class are:

['students' 'school' 'learning' 'classroom' 'not' 'learn' 'help' 'price'
 'many' 'nannan' 'need' 'work' 'reading' 'use' 'love'
 'teacher_number_of_previously_posted_projects' 'day' 'able' 'come'
 'class'] [7889 7451 6154 4472 6756 6097 5719 8878 6655 6443 6669 8672 4566 8879
 6333 3875 7259 8479 4803 6489]

The top 20 features from the negative class are :

['students' 'school' 'learning' 'classroom' 'not' 'learn' 'help' 'price'
 'nannan' 'many' 'need' 'work' 'come'
 'teacher_number_of_previously_posted_projects' 'love' 'able' 'reading'
 'use' 'day' 'materials'] [7889 7451 6154 4472 6756 6097 5719 8878 6655 6443 6669 8672 4566 8879
 6333 3875 7259 8479 4803 6489]
```

2.4.2 Applying NB on TFIDF, SET 2

Encoding eassay and project_title using TFIDF

In [167]:
```python
# 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)

print("="*100)

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer.transform(X_train['preprocessed_essays'].values)
X_cv_essay_tfidf = vectorizer.transform(X_cv['preprocessed_essays'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['preprocessed_essays'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)

#Encoding preprocessed title

ectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['preprocessed_title'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_tfidf = vectorizer.transform(X_train['preprocessed_title'].values)
X_cv_title_tfidf = vectorizer.transform(X_cv['preprocessed_title'].values)
X_test_title_tfidf = vectorizer.transform(X_test['preprocessed_title'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print("="*100)
```

```
(44890, 21) (44890,)
(22110, 21) (22110,)
(33000, 21) (33000,)
====================================================================================================
After vectorizations
(44890, 11632) (44890,)
(22110, 11632) (22110,)
(33000, 11632) (33000,)
====================================================================================================
After vectorizations
(44890, 2005) (44890,)
(22110, 2005) (22110,)
(33000, 2005) (33000,)
====================================================================================================
```

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

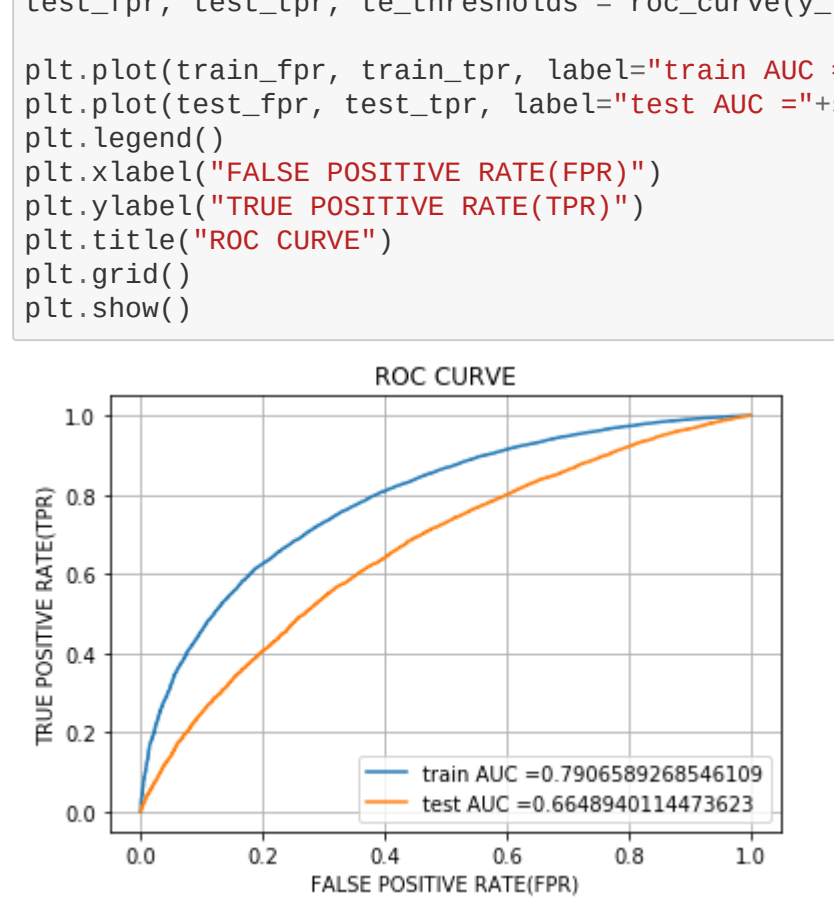
In [168]:
```python
# preparing Set2
# we need to merge all the numerical vectors i.e catogorical, text, numerical vectors
from scipy.sparse import hstack
X_train_set2 = hstack((X_train_title_tfidf, X_train_essay_tfidf, X_train_categories_one_hot, X_train_sub_categories_one_hot, X_train_school_state_one_hot, X_train_teacher_prefix_one_hot, X_train_project_grade_category_one_hot, X_train_price_norm,X_train_teacher_number_of_previously_posted_projects_norm)).tocsr()
X_cv_set2= hstack((X_cv_title_tfidf, X_cv_essay_tfidf, X_cv_categories_one_hot, X_cv_sub_categories_one_hot, X_cv_school_state_one_hot, X_cv_teacher_prefix_one_hot, X_cv_project_grade_category_one_hot, X_cv_price_norm, X_cv_teacher_number_of_previously_posted_projects_norm)).tocsr()
X_test_set2 = hstack((X_test_title_tfidf, X_test_essay_tfidf, X_test_categories_one_hot, X_test_sub_categories_one_hot, X_test_school_state_one_hot, X_test_teacher_prefix_one_hot, X_test_project_grade_category_one_hot, X_test_price_norm, X_test_teacher_number_of_previously_posted_projects_norm)).tocsr()

print("Final Data matrix")
print(X_train_set2.shape, y_train.shape)
print(X_cv_set2.shape, y_cv.shape)
print(X_test_set2.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(44890, 13738) (44890,)
(22110, 13738) (22110,)
(33000, 13738) (33000,)
====================================================================================================
```

Hyperparameter tuning

In [169]:
```python
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or non-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
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_set2, y_train)

    y_train_pred = batch_predict(model, X_train_set2)
    y_cv_pred = batch_predict(model, X_cv_set2)

    # 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))

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

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

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

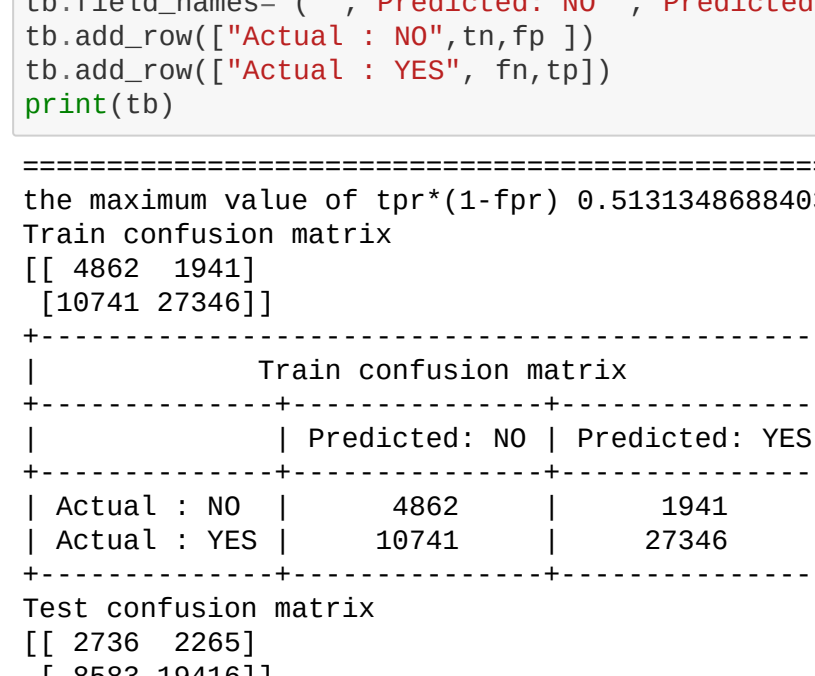
```
100%|██████████| 6/6 [00:01<00:00,  4.34it/s]
```



Testing the performance of the model on test data, plotting ROC Curves

In [170]:
```python
#here we are choosing best value of alpha based on for loop results
best_alpha = 0.1
```

In [171]:
```python
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


model = MultinomialNB(alpha=best_alpha)
model.fit(X_train_set2, 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(model, X_train_set2)
y_test_pred = batch_predict(model, X_test_set2)

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

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("FALSE POSITIVE RATE(FPR)")
plt.ylabel("TRUE POSITIVE RATE(TPR)")
plt.title("ROC CURVE")
plt.grid()
plt.show()
```



Confusion Matrix

In [172]:
```python
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
tn,fp,fn,tp=confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)).ravel()

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

print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
tn,fp,fn,tp=confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)).ravel()
tb = PrettyTable()
tb.title= 'Test confusion matrix'
tb.field_names= ("","Predicted: NO" ,"Predicted: YES")
tb.add_row(["Actual : NO",tn,fp ])
tb.add_row(["Actual : YES", fn,tp])
print(tb)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.5131348688403463 for threshold 0.847
Train confusion matrix
[[ 4862  1941]
 [10741 27346]]
+----------------------------------------------+
|             Train confusion matrix           |
+--------------+---------------+---------------+
|              | Predicted: NO | Predicted: YES |
+--------------+---------------+---------------+
| Actual : NO  |      4862     |      1941      |
| Actual : YES |     10741     |     27346      |
+--------------+---------------+---------------+
Test confusion matrix
[[ 2736  2265]
 [ 8583 19416]]
+----------------------------------------------+
|             Test confusion matrix            |
+--------------+---------------+---------------+
|              | Predicted: NO | Predicted: YES |
+--------------+---------------+---------------+
| Actual : NO  |      2736     |      2265      |
| Actual : YES |      8583     |     19416      |
+--------------+---------------+---------------+
```

2. Summary

Comparing all models performance using Prettytable library

In [174]:
```python
#ref link: http://zetcode.com/python/prettytable/
tb = PrettyTable()
tb.title= 'Summary'
tb.field_names= ("Vectorizer","Model" ,"Hyperparameter", "AUC")
tb.add_row(["BOW", "Naive Bayes",0.1, 0.6852])
tb.add_row(["TFIDF", "Naive Bayes",0.1, 0.6648 ])
print(tb)
```

```
+---------------------------------------------------+
|                      Summary                      |
+------------+-------------+----------------+--------+
| Vectorizer |    Model    | Hyperparameter |  AUC   |
+------------+-------------+----------------+--------+
|    BOW     | Naive Bayes |      0.1       | 0.6852 |
|   TFIDF    | Naive Bayes |      0.1       | 0.6648 |
+------------+-------------+----------------+--------+
```

Observation:
- We observed better AUC value i.e 0.6852 using Bow vectorizer as compare to using TFIDF vectorizer.

In [ ]: