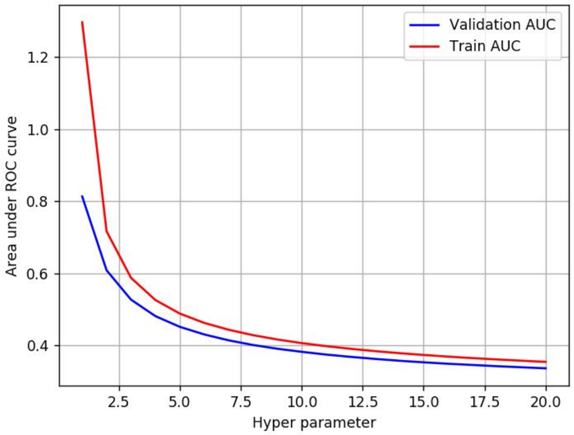
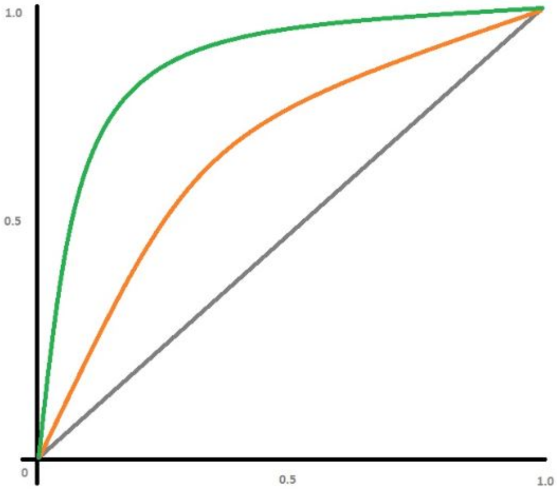
Assignment 6: Apply NB

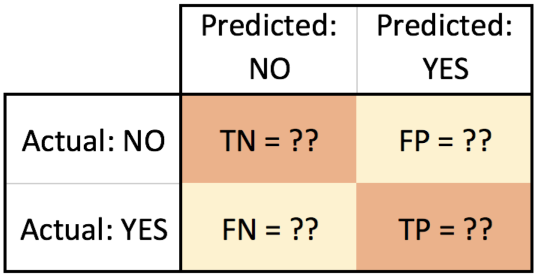
1. **Apply Multinomial NB on these feature sets**
   - Set 1: categorical, numerical features + preprocessed_eassay (BOW)
   - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)
2. **The hyper paramter tuning(find best alpha:smoothing parameter)**
   - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) 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.

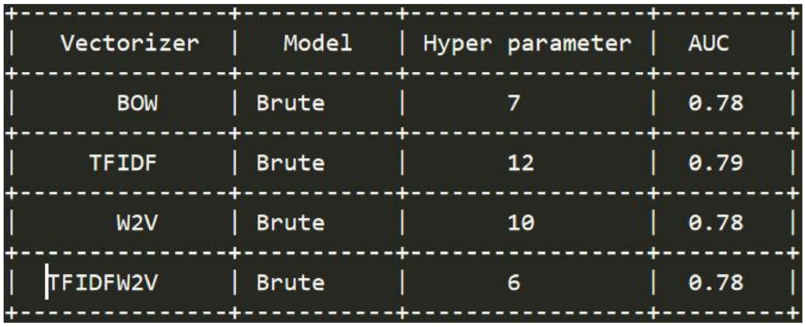


- Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points



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

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 (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   |
+---------------+-----------+-----------------+---------+
```

2. Naive Bayes

1.1 Loading Data

In [149]:

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

import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
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
# Tutorial about Python regular expressions: https://pymotw.com/2/re/

import pickle
from tqdm import tqdm
import os
from collections import Counter
```

In [150]:

```python
import pandas as pd
# data = pandas.read_csv('preprocessed_data.csv')
data  = pd.read_csv('preprocessed_data.csv', nrows=50000)

```

In [151]:

```python
data.head(1)
```

Out[151]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projec
0	ca	mrs	grades_prek_2	

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

In [152]:

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

In [153]:

```
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)
X.head(1)
```

Out[153]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projec
0	ca	mrs	grades_prek_2	

In [154]:

```
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, str
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0

print(X_train.shape)
print(X_test.shape)
print("***************************")
print(y_train.shape)
print(y_test.shape)
```

```
(22445, 8)
(16500, 8)
***************************
(22445,)
(16500,)
```

1.3 Make Data Model Ready: encoding eassay, and project_title

BOW Vectorization

In [155]:

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

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

BOW Vectorization Eassy

In [156]:

```python
from sklearn.feature_extraction.text import CountVectorizer

vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['essay'].values)
X_train_essay_bow = vect.transform(X_train['essay'].values)
X_cv_essay_bow = vect.transform(X_cv['essay'].values)
X_test_essay_bow = vect.transform(X_test['essay'].values)
essay_bow_features=vect.get_feature_names()

print('X_train_essay_bow.shape',X_train_essay_bow.shape, y_train.shape)
print('X_test_essay_bow.shape',X_test_essay_bow.shape, y_test.shape)
print('X_cv_essay_bow.shape',X_cv_essay_bow.shape, y_cv.shape)
print(essay_bow_features[:10])
```

```
X_train_essay_bow.shape (22445, 50000) (22445,)
X_test_essay_bow.shape (16500, 50000) (16500,)
X_cv_essay_bow.shape (11055, 50000) (11055,)
['00', '00 pm', '000', '000 students', '10', '10 000', '10 11', '10 11 year',
'10 15', '10 girls']
```

1.4 Make Data Model Ready: encoding numerical, categorical features

In [157]:

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

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

school_state -OHE

In [158]:

```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['school_state'].values)
X_train_ohe_school_state= vect.transform(X_train['school_state'].values)
X_cv_ohe_school_state= vect.transform(X_cv['school_state'].values)
X_test_ohe_school_state= vect.transform(X_test['school_state'].values)
school_state_features=vect.get_feature_names()

print('X_train_ohe_school_state',X_train_ohe_school_state.shape, y_train.shap
print('X_test_ohe_school_state',X_test_ohe_school_state.shape, y_test.shape)
print('X_cv_ohe_school_state',X_cv_ohe_school_state.shape, y_cv.shape)
print(school_state_features[:10])
```

```
X_train_ohe_school_state (22445, 50) (22445,)
X_test_ohe_school_state (16500, 50) (16500,)
X_cv_ohe_school_state (11055, 50) (11055,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl']
```

teacher_prefix -OHE

In [159]:

```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['teacher_prefix'].values)
X_train_ohe_teacher_prefix= vect.transform(X_train['teacher_prefix'].values)
X_cv_ohe_teacher_prefix= vect.transform(X_cv['teacher_prefix'].values)
X_test_ohe_teacher_prefix= vect.transform(X_test['teacher_prefix'].values)
teacher_prefix_features=vect.get_feature_names()

print('X_train_ohe_teacher_prefix',X_train_ohe_teacher_prefix.shape, y_train.
print('X_test_ohe_teacher_prefix',X_test_ohe_teacher_prefix.shape, y_test.sha
print('X_cv_ohe_teacher_prefix',X_cv_ohe_teacher_prefix.shape, y_cv.shape)
print(teacher_prefix_features[:10])
```

```
X_train_ohe_teacher_prefix (22445, 4) (22445,)
X_test_ohe_teacher_prefix (16500, 4) (16500,)
X_cv_ohe_teacher_prefix (11055, 4) (11055,)
['mr', 'mrs', 'ms', 'teacher']
```

project_grade_category-OHE

In [160]:

```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['project_grade_category'].values)
X_train_ohe_project_grade_category= vect.transform(X_train['project_grade_cat
X_cv_ohe_project_grade_category= vect.transform(X_cv['project_grade_category'
X_test_ohe_project_grade_category= vect.transform(X_test['project_grade_categ
project_grade_category_features=vect.get_feature_names()

print('X_train_ohe_project_grade_category',X_train_ohe_project_grade_category
print('X_test_ohe_project_grade_category',X_test_ohe_project_grade_category.s
print('X_cv_ohe_project_grade_category',X_cv_ohe_project_grade_category.shape
print(project_grade_category_features[:10])

```

```
X_train_ohe_project_grade_category (22445, 4) (22445,)
X_test_ohe_project_grade_category (16500, 4) (16500,)
X_cv_ohe_project_grade_category (11055, 4) (11055,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
```

clean_categories-OHE

In [161]:

```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['clean_categories'].values)
X_train_ohe_clean_categories= vect.transform(X_train['clean_categories'].valu
X_cv_ohe_clean_categories= vect.transform(X_cv['clean_categories'].values)
X_test_ohe_clean_categories= vect.transform(X_test['clean_categories'].values
clean_categories_features=vect.get_feature_names()

print('X_train_ohe_clean_categories',X_train_ohe_clean_categories.shape, y_tr
print('X_test_ohe_clean_categories',X_test_ohe_clean_categories.shape, y_test
print('X_cv_ohe_clean_categories',X_cv_ohe_clean_categories.shape, y_cv.shape
print(clean_categories_features[:10])

```

```
X_train_ohe_clean_categories (22445, 36) (22445,)
X_test_ohe_clean_categories (16500, 36) (16500,)
X_cv_ohe_clean_categories (11055, 36) (11055,)
['appliedlearning', 'appliedlearning health_sports', 'appliedlearning history_c
ivics', 'appliedlearning literacy_language', 'appliedlearning math_science', 'a
ppliedlearning music_arts', 'appliedlearning specialneeds', 'health_sports', 'h
ealth_sports appliedlearning', 'health_sports literacy_language']
```

clean_subcategories-OHE

In [162]:

```
vect= CountVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['clean_subcategories'].values)
X_train_ohe_clean_subcategories= vect.transform(X_train['clean_subcategories'
X_cv_ohe_clean_subcategories= vect.transform(X_cv['clean_subcategories'].valu
X_test_ohe_clean_subcategories= vect.transform(X_test['clean_subcategories'].
clean_subcategories_features=vect.get_feature_names()

print('X_train_ohe_clean_subcategories',X_train_ohe_clean_subcategories.shape
print('X_test_ohe_clean_subcategories',X_test_ohe_clean_subcategories.shape,
print('X_cv_ohe_clean_subcategories',X_cv_ohe_clean_subcategories.shape, y_cv
print(clean_subcategories_features[:10])
```

```
X_train_ohe_clean_subcategories (22445, 147) (22445,)
X_test_ohe_clean_subcategories (16500, 147) (16500,)
X_cv_ohe_clean_subcategories (11055, 147) (11055,)
['appliedsciences', 'appliedsciences charactereducation', 'appliedsciences coll
ege_careerprep', 'appliedsciences earlydevelopment', 'appliedsciences environme
ntalscience', 'appliedsciences esl', 'appliedsciences extracurricular', 'applie
dsciences health_lifescience', 'appliedsciences history_geography', 'appliedsci
ences literacy']
```

Normalize numerical features

price

In [163]:

```
from sklearn.preprocessing import Normalizer
norml=Normalizer()
norml.fit(X_train['price'].values.reshape(1,-1))
X_train_norml_price=norml.transform(X_train['price'].values.reshape(1,-1))
X_test_norml_price=norml.transform(X_test['price'].values.reshape(1,-1))
X_cv_norml_price=norml.transform(X_cv['price'].values.reshape(1,-1))

#X_cv_norm_price=norm.transform(xcv['price'].values.reshape(-1,1))
print('X_train_norml_price shape',X_train_norml_price.shape,y_train.shape)
print('X_test_norml_price shape',X_test_norml_price.shape,y_test.shape)
print('X_cv_norml_price shape',X_cv_norml_price.shape,y_cv.shape)

```

```
X_train_norml_price shape (1, 22445) (22445,)
X_test_norml_price shape (1, 16500) (16500,)
X_cv_norml_price shape (1, 11055) (11055,)
```

In [ ]:

```

```

teacher_number_of_previously_posted_projects

In [164]:

```python
norml.fit(X_train['teacher_number_of_previously_posted_projects'].values.resh
X_train_norml_teacher_number_of_previously_posted_projects=norml.transform(X_
X_test_norml_teacher_number_of_previously_posted_projects=norml.transform(X_t
X_cv_norml_teacher_number_of_previously_posted_projects=norml.transform(X_cv[


#X_cv_norm_price=norm.transform(xcv['price'].values.reshape(-1,1))
print('X_train_norml_teacher_number_of_previously_posted_projects shape',X_tr
print('X_test_norml_teacher_number_of_previously_posted_projects shape',X_tes
#print('X_cv_norm_price shape',xcv_norm_price.shape,ycv.shape)
print('X_cv_norml_teacher_number_of_previously_posted_projects shape',X_cv_nc
#print('X_cv_norm_price shape',xcv_norm_price.shape,ycv.shape)
```

```
X_train_norml_teacher_number_of_previously_posted_projects shape (1, 22445) (22
445,)
X_test_norml_teacher_number_of_previously_posted_projects shape (1, 16500) (165
00,)
X_cv_norml_teacher_number_of_previously_posted_projects shape (1, 11055) (1105
5,)
```

SET 1

In [165]:

```python
#  https://stackoverflow.com/a/19710648/4084039
# combine all features into one single set

X_train_norml_tnoprepst_projects=X_train_norml_teacher_number_of_previously_p
X_test_norml_tnoprepst_projects=X_test_norml_teacher_number_of_previously_pos
X_cv_norml_tnoprepst_projects=X_cv_norml_teacher_number_of_previously_posted_

X_train_norml_price= X_train_norml_price.reshape(22445,1)
X_test_norml_price=X_test_norml_price.reshape(16500,1)
X_cv_norml_price=X_cv_norml_price.reshape(11055,1)

from scipy.sparse import hstack
X_tr_set1 = hstack((X_train_essay_bow,X_train_norml_price,X_train_norml_tnopr
X_tst_set1= hstack((X_test_essay_bow,X_test_norml_price,X_test_norml_tnopreps
X_cr_set1= hstack((X_cv_essay_bow,X_cv_norml_price,X_cv_norml_tnoprepst_proje


print("Final Data matrix")
print(X_tr_set1.shape, y_train.shape)
print(X_cr_set1.shape, y_cv.shape)
print(X_tst_set1.shape, y_test.shape)

```

```
Final Data matrix
(22445, 50243) (22445,)
(11055, 50243) (11055,)
(16500, 50243) (16500,)
```

TFIDF vectorization

Eassy

In [166]:

```
vect= TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=50000)
vect.fit(X_train['essay'].values)
X_train_essay_tfidf = vect.transform(X_train['essay'].values)
X_cv_essay_tfidf = vect.transform(X_cv['essay'].values)
X_test_essay_tfidf = vect.transform(X_test['essay'].values)
essay_tfidf_features=vect.get_feature_names()

print('X_train_essay_tfidf.shape',X_train_essay_tfidf.shape, y_train.shape)
print('X_test_essay_tfidf.shape',X_test_essay_tfidf.shape, y_test.shape)
print('X_cv_essay_tfidf.shape',X_cv_essay_tfidf.shape, y_cv.shape)
# print('X_cv_essay_bow.shape',X_cv_essay_bow.shape, y_cv.shape)
```

```
X_train_essay_tfidf.shape (22445, 50000) (22445,)
X_test_essay_tfidf.shape (16500, 50000) (16500,)
X_cv_essay_tfidf.shape (11055, 50000) (11055,)
```

SET 2

In [167]:

```
#  https://stackoverflow.com/a/19710648/4084039
# combine all features into one single set

X_train_norml_tnoprepst_projects=X_train_norml_teacher_number_of_previously_p
X_test_norml_tnoprepst_projects=X_test_norml_teacher_number_of_previously_pos
X_cv_norml_tnoprepst_projects=X_cv_norml_teacher_number_of_previously_posted_

X_train_norml_price= X_train_norml_price.reshape(22445,1)
X_test_norml_price=X_test_norml_price.reshape(16500,1)
X_cv_norml_price=X_cv_norml_price.reshape(11055,1)

from scipy.sparse import hstack
X_tr_set2 = hstack((X_train_essay_tfidf,X_train_norml_price,X_train_norml_tno
X_tst_set2= hstack((X_test_essay_tfidf,X_test_norml_price,X_test_norml_tnopre
X_cr_set2= hstack((X_cv_essay_tfidf,X_cv_norml_price,X_cv_norml_tnoprepst_pro


print("Final Data matrix")
print(X_tr_set2.shape, y_train.shape)
print(X_cr_set2.shape, y_cv.shape)
print(X_tst_set2.shape, y_test.shape)

```

```
Final Data matrix
(22445, 50243) (22445,)
(11055, 50243) (11055,)
(16500, 50243) (16500,)
```

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

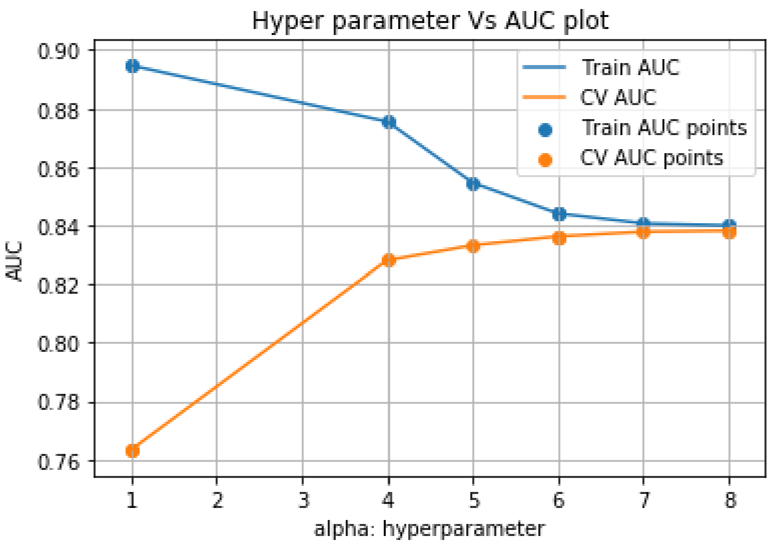
In [228]:

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

FOR SET 1

In [229]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.G
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
from sklearn import cross_validation
import math
import warnings
warnings.filterwarnings("ignore")


model=MultinomialNB()
parameters = {'alpha': sp_randint(0.1,10)}
clf = RandomizedSearchCV(estimator = model,param_distributions = parameters,
clf.fit(X_tr_set1, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_alpha'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
alpha=  results['param_alpha']

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

plt.plot(alpha, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2

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("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
results.head()
```



Out[229]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_
7	0.128674	0.000943	0.016334	0.000472	1	{'alpha': 1}	0.7
8	0.139341	0.003682	0.021668	0.003772	1	{'alpha': 1}	0.7
1	0.130674	0.002357	0.019334	0.004714	4	{'alpha': 4}	0.8
3	0.143675	0.006600	0.018334	0.002495	5	{'alpha': 5}	0.8
4	0.159676	0.026839	0.018001	0.002160	6	{'alpha': 6}	0.8

In [230]:

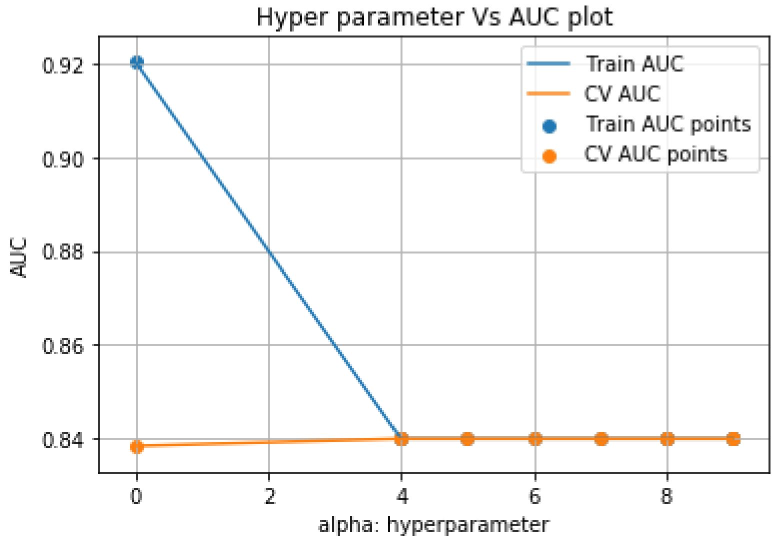
```
best_alpha_set1 =clf.best_estimator_
print(best_alpha_set1.alpha)
```

```
8
```

FOR SET 2

In [231]:

```
model=MultinomialNB()
parameters = {'alpha': sp_randint(0.1,10)}
clf = RandomizedSearchCV(estimator = model,param_distributions = parameters,
clf.fit(X_tr_set2, y_train)

results = pd.DataFrame.from_dict(clf.cv_results_)
results = results.sort_values(['param_alpha'])

train_auc= results['mean_train_score']
train_auc_std= results['std_train_score']
cv_auc = results['mean_test_score']
cv_auc_std= results['std_test_score']
alpha=  results['param_alpha']

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

plt.plot(alpha, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2

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("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
results.head()
```



Out[231]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_
2	0.131341	0.001247	0.016001	1.123916e-07	0	{'alpha': 0}	0.8
0	0.133674	0.003300	0.016668	9.427407e-04	4	{'alpha': 4}	0.8
3	0.132008	0.001414	0.016334	4.714266e-04	4	{'alpha': 4}	0.8
9	0.146342	0.010531	0.019001	3.559326e-03	5	{'alpha': 5}	0.8
5	0.130007	0.001633	0.015668	4.714827e-04	6	{'alpha': 6}	0.8

In [232]:

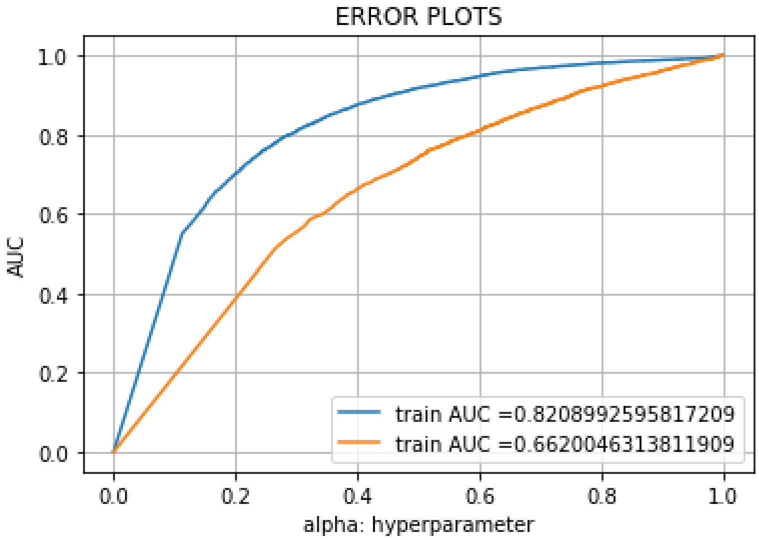
```
best_alpha_set2 =clf.best_estimator_
# best_alpha_set2.best_alpha
print(best_alpha_set2.alpha)
```

```
4
```

AUC OF SET 1- BOW

In [233]:

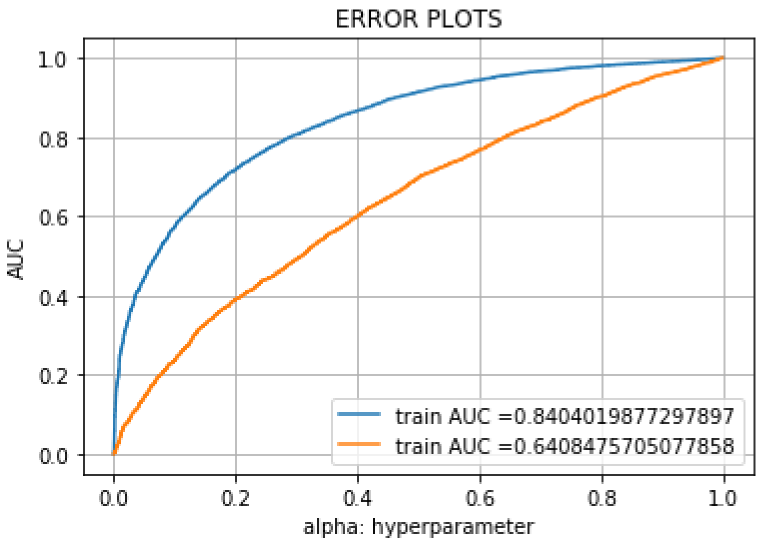
```python
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
    # not the predicted outputs

    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 - 490
    # 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

clf_set1=MultinomialNB(best_alpha_set1.alpha)
clf_set1.fit(X_tr_set1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability esti
# not the predicted outputs

y_train_pred = batch_predict(clf_set1, X_tr_set1)
y_test_pred = batch_predict(clf_set1, X_tst_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)

auc_set1=auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



AUC OF SET 2- TFIDF

In [234]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
    # 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 - 490
    # 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

clf_set2=MultinomialNB(best_alpha_set2.alpha)
clf_set2.fit(X_tr_set1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability esti
# not the predicted outputs

y_train_pred = batch_predict(clf_set2, X_tr_set2)
y_test_pred = batch_predict(clf_set2, X_tst_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)

auc_set2=auc(test_fpr, test_tpr)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr))
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Confusion matrix

In [235]:

```python
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 h
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshol
    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

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)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.5771535030601741 for threshold 0.847
Train confusion matrix
[[ 2828   767]
 [ 5020 13830]]
Test confusion matrix
[[1297 1345]
 [4092 9766]]
```

TOP 20 Features

In [236]:

```python
import numpy as np
feature=[]
feature.extend(essay_bow_features)
feature.extend(school_state_features)
feature.extend(teacher_prefix_features)
feature.extend(project_grade_category_features)
feature.extend(clean_categories_features)
feature.extend(clean_subcategories_features)
feature.extend(['price'])
feature.extend(['teacher_number_of_previously_posted_projects'])

print(len(feature))
```

```
50243
```

In [237]:

```
lst0=np.argsort((clf_set1.feature_log_prob_)[0])[:20]
top_20_features_0=np.take(feature,lst0)
print(top_20_features_0)
print('#######################################################')
lst1=np.argsort((clf_set1.feature_log_prob_)[1])[:20]
top_20_features_1=np.take(feature,lst1)
print(top_20_features_1)
```

```
['challenge find' 'scooter' 'hokki stools give' 'creative group'
 'they represent' 'hokki stools wobble' 'hokki stools would'
 'hokki stools would allow' 'scope magazine' 'scientific calculators'
 'science unit' 'sugary' 'many students special needs' 'creative active'
 'home important' 'science social studies topics' 'allow students listen'
 'holder' 'become great readers' 'creative hard']
#######################################################
['literacy_language specialneeds' 'students moderate severe disabilities'
 'comfortable these' 'thinking this simply' 'go we need help'
 'classroom despite challenges face' 'one effective' 'highest quality'
 'thinking this simply cannot' 'functional life' 'supplies donated'
 'this simply cannot done' 'as small' 'this simply cannot' 'punishment'
 'simply cannot done kids' 'learning sight' 'solve problems make choices'
 'problems make choices' 'civics_government literature_writing']
```

3. Summary

as mentioned in the step 5 of instructions

In [238]:

```
! pip install Prettytable
```

```
Requirement already satisfied: Prettytable in c:\users\aaa\anaconda3\lib\site-p
ackages (0.7.2)

distributed 1.21.8 requires msgpack, which is not installed.
You are using pip version 10.0.1, however version 20.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' com
mand.
```

In [239]:

```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
pryt=PrettyTable()
pryt.field_names = ["Vectorizer", "Model", "Hyper-Parameter", "AUC"]
pryt.add_row([" BOW ", "Naive Bayes", best_alpha_set1.alpha,auc_set1])
pryt.add_row([" TFIDF ", "Naive Bayes", best_alpha_set2.alpha, auc_set2])
print(pryt)
```

```
+------------+-------------+-----------------+--------------------+
| Vectorizer |    Model    | Hyper-Parameter |        AUC         |
+------------+-------------+-----------------+--------------------+
|    BOW     | Naive Bayes |        8        | 0.6620046313811909 |
|   TFIDF    | Naive Bayes |        4        | 0.6408475705077858 |
+------------+-------------+-----------------+--------------------+
```

In [ ]:

```
```