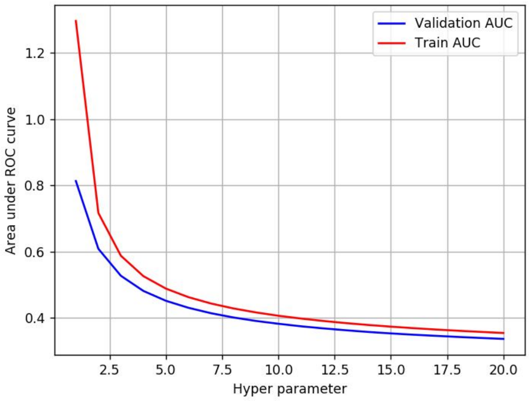
Assignment 6: Apply NB

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

    - Features that need to be considered
      essay
      while encoding essay, try to experiment with the max_features and n_grams parameter of vectorizers and see if it increases AUC score.
      categorical features
      - teacher_prefix
      - project_grade_category
      - school_state
      - clean_categories
      - clean_subcategories
      numerical features
      - price
      - teacher_number_of_previously_posted_projects
      while encoding the numerical features check [this](#) and [this](#)

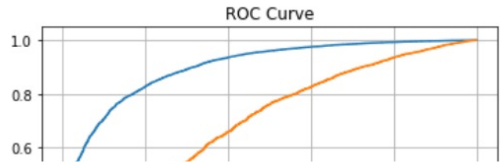
    - Set 1: categorical, numerical features + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)
6. **The hyper paramter tuning(find best alpha:smoothing parameter)**

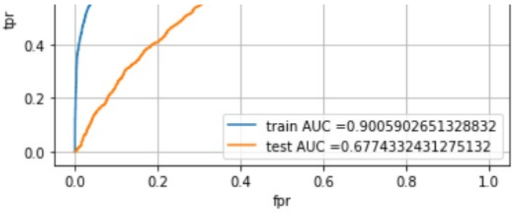
    - Consider alpha values in range: 10^-5 to 10^2 like [0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
    - Explore class_prior = [0.5, 0.5] parameter which can be present in MultinomialNB function(go through [this](#)) then check how results might change.
    - Find the best hyper parameter which will give the maximum [AUC](#) value
    - For hyper parameter tuning using k-fold cross validation(use GridsearchCV or RandomsearchCV)/simple cross validation data (write for loop to iterate over hyper parameter values)
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



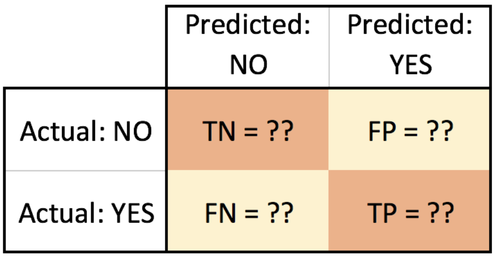
  -while plotting take log(alpha) on your X-axis so that it will be more readable

    - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.





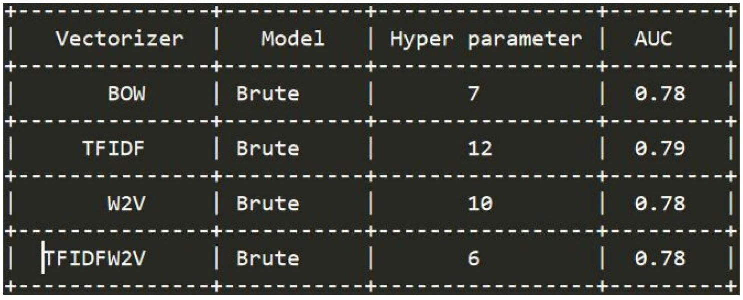
- **Along with plotting ROC curve, you need to print the [confusion matrix](#) with predicted and original labels of test data points**



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

-plot the confusion matrix in heatmaps, while plotting the confusion matrix go through the [link](#)

7. **find the top 20 features from either from feature Set 1 or feature Set 2 using values of \`feature_log_prob_\` parameter of \`MultinomialNB\` (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print BOTH positive as well as negative corresponding feature names.**
   - go through the [link](#)
8. **You need to summarize the results at the end of the notebook, summarize it in the table format**



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

In [62]:

```
%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
from sklearn.preprocessing import Normalizer
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import math
import pickle
from tqdm import tqdm
import os
from scipy.sparse import hstack
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import seaborn as sns
import matplotlib.pyplot as plt
from prettytable import PrettyTable
```

2. Naive Bayes

1.1 Loading Data

In [69]:

```
import pandas
data = pandas.read_csv('/content/drive/MyDrive/temp/naive bayes/preprocessed_data.csv', n
rows=100000) #using 100k points data
data.head(5)

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

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

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

In [70]:

```
# please write all the code with proper documentation, and proper titles for each subsect
ion
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your c
ode
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
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) #t
rain-test splitting
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, strati
fy=y_train)#train cv splitting
print(len(X_train))
print(len(X_test))
print(len(y_train))
print(len(y_test))
print(len(X_cv))
print(len(y_cv))
```

```
44890
33000
44890
33000
22110
22110
```

1.3 Make Data Model Ready: encoding eassay, and project_title

In [71]:

```
# please write all the code with proper documentation, and proper titles for each subsect
ion
```

```
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your c
ode
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
preprocessed_train_essays = X_train['essay'].values
preprocessed_test_essays = X_test['essay'].values

# We are considering only the words which appeared in at least 10 documents(rows or proje
cts).
vectorizer = CountVectorizer(min_df=10) #initiating count vectorizer
train_bow = vectorizer.fit_transform(preprocessed_train_essays)
cv_bow=vectorizer.transform(X_cv['essay'].values)
test_bow=vectorizer.transform(preprocessed_test_essays)
print("Shape of matrix after one hot encodig train",train_bow.shape)
print("Shape of matrix after one hot encodig test ",test_bow.shape)
print("Shape of matrix after one hot encodig test ",cv_bow.shape)

vectorizer1 = TfidfVectorizer(min_df=10)#initiating tfidf vectorizer
train_tfidf = vectorizer1.fit_transform(preprocessed_train_essays)
cv_tfidf=vectorizer1.transform(X_cv['essay'].values)
test_tfidf=vectorizer1.transform(preprocessed_test_essays)
print("Shape of matrix after one hot encodig ",train_tfidf.shape)
print("Shape of matrix after one hot encodig ",test_tfidf.shape)
print("Shape of matrix after one hot encodig ",cv_tfidf.shape)
```

```
Shape of matrix after one hot encodig train (44890, 11706)
Shape of matrix after one hot encodig test  (33000, 11706)
Shape of matrix after one hot encodig test  (22110, 11706)
Shape of matrix after one hot encodig  (44890, 11706)
Shape of matrix after one hot encodig  (33000, 11706)
Shape of matrix after one hot encodig  (22110, 11706)
```

1.4 Make Data Model Ready: encoding numerical, categorical features

In [72]:

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

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


preprocessed_train_school_state = X_train['school_state'].values
preprocessed_test_school_state = X_test['school_state'].values


school_state_ohe_train = vectorizer.fit_transform(preprocessed_train_school_state)
school_state_ohe_cv = vectorizer.transform(X_cv['school_state'].values)
school_state_ohe_test = vectorizer.transform(preprocessed_test_school_state)
print("Shape of matrix after one hot encodig school_state_ohe_train",school_state_ohe_tra
in.shape)
print("Shape of matrix after one hot encodig school_state_ohe_cv",school_state_ohe_cv.sha
pe)
print("Shape of matrix after one hot encodig school_state_ohe_test",school_state_ohe_test
.shape)


preprocessed_train_teacher_prefix = X_train['teacher_prefix'].values
preprocessed_test_teacher_prefix = X_test['teacher_prefix'].values


teacher_prefix_ohe_train = vectorizer.fit_transform(preprocessed_train_teacher_prefix)
teacher_prefix_ohe_cv = vectorizer.transform(X_cv['teacher_prefix'].values)
teacher_prefix_ohe_test = vectorizer.transform(preprocessed_test_teacher_prefix)
print("Shape of matrix after one hot encodig teacher_prefix_ohe_train",teacher_prefix_ohe_train.shape)
print("Shape of matrix after one hot encodig teacher_prefix_ohe_cv",teacher_prefix_ohe_cv.shape)
print("Shape of matrix after one hot encodig teacher_prefix_ohe_test",teacher_prefix_ohe_test.shape)



preprocessed_train_project_grade_category = X_train['project_grade_category'].values
preprocessed_test_project_grade_category = X_test['project_grade_category'].values


project_grade_category_ohe_train = vectorizer.fit_transform(preprocessed_train_project_grade_category)
project_grade_category_ohe_cv = vectorizer.transform(X_cv['project_grade_category'].values)
project_grade_category_ohe_test = vectorizer.transform(preprocessed_test_project_grade_category)
print("Shape of matrix after one hot encodig project_grade_category_ohe_train",project_grade_category_ohe_train.shape)
print("Shape of matrix after one hot encodig project_grade_category_ohe_cv",project_grade_category_ohe_cv.shape)
print("Shape of matrix after one hot encodig project_grade_category_ohe_test",project_grade_category_ohe_test.shape)

preprocessed_train_clean_categories = X_train['clean_categories'].values
preprocessed_test_clean_categories = X_test['clean_categories'].values


clean_categories_ohe_train = vectorizer.fit_transform(preprocessed_train_clean_categories)
clean_categories_ohe_cv = vectorizer.transform(X_cv['clean_categories'].values)
clean_categories_ohe_test = vectorizer.transform(preprocessed_test_clean_categories)
print("Shape of matrix after one hot encodig clean_categories_ohe_train",clean_categories_ohe_train.shape)
print("Shape of matrix after one hot encodig clean_categories_ohe_cv",clean_categories_ohe_cv.shape)
print("Shape of matrix after one hot encodig clean_categories_ohe_test",clean_categories_ohe_test.shape)

preprocessed_train_clean_subcategories = X_train['clean_subcategories'].values
preprocessed_test_clean_subcategories = X_test['clean_subcategories'].values


clean_subcategories_ohe_train = vectorizer.fit_transform(preprocessed_train_clean_subcategories)
clean_subcategories_ohe_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
clean_subcategories_ohe_test = vectorizer.transform(preprocessed_test_clean_categories)
print("Shape of matrix after one hot encodig clean_subcategories_ohe_train ",clean_subcategories_ohe_train.shape)
print("Shape of matrix after one hot encodig clean_subcategories_ohe_cv",clean_subcategories_ohe_cv.shape)
print("Shape of matrix after one hot encodig  clean_subcategories_ohe_test",clean_subcategories_ohe_test.shape)




normalizer = Normalizer()
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("X_train_price_norm:",len(X_train_price_norm))
print("X_cv_price_norm",len(X_cv_price_norm))
print("X_test_price_norm",len(X_test_price_norm))


#hstack using BOW in essay feature
X_tr = hstack((train_bow, school_state_ohe_train, teacher_prefix_ohe_train, project_grade
_category_ohe_train, X_train_price_norm)).tocsr()
X_cr = hstack((cv_bow, school_state_ohe_cv, teacher_prefix_ohe_cv, project_grade_categor
y_ohe_cv, X_cv_price_norm)).tocsr()
X_te = hstack((test_bow, school_state_ohe_test, teacher_prefix_ohe_test, project_grade_ca
tegory_ohe_test, X_test_price_norm)).tocsr()

print("Final Data matrix BOW")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Shape of matrix after one hot encodig school_state_ohe_train (44890, 51)
Shape of matrix after one hot encodig school_state_ohe_cv (22110, 51)
Shape of matrix after one hot encodig school_state_ohe_test (33000, 51)
Shape of matrix after one hot encodig teacher_prefix_ohe_train (44890, 4)
Shape of matrix after one hot encodig teacher_prefix_ohe_cv (22110, 4)
Shape of matrix after one hot encodig teacher_prefix_ohe_test (33000, 4)
Shape of matrix after one hot encodig project_grade_category_ohe_train (44890, 4)
Shape of matrix after one hot encodig project_grade_category_ohe_cv (22110, 4)
Shape of matrix after one hot encodig project_grade_category_ohe_test (33000, 4)
Shape of matrix after one hot encodig clean_categories_ohe_train (44890, 9)
Shape of matrix after one hot encodig clean_categories_ohe_cv (22110, 9)
Shape of matrix after one hot encodig clean_categories_ohe_test (33000, 9)
Shape of matrix after one hot encodig clean_subcategories_ohe_train  (44890, 30)
Shape of matrix after one hot encodig clean_subcategories_ohe_cv (22110, 30)
Shape of matrix after one hot encodig  clean_subcategories_ohe_test (33000, 30)
X_train_price_norm: 44890
X_cv_price_norm 22110
X_test_price_norm 33000
Final Data matrix BOW
(44890, 11766) (44890,)
(22110, 11766) (22110,)
(33000, 11766) (33000,)
========================================================================================
============
```

SET1:BOW

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

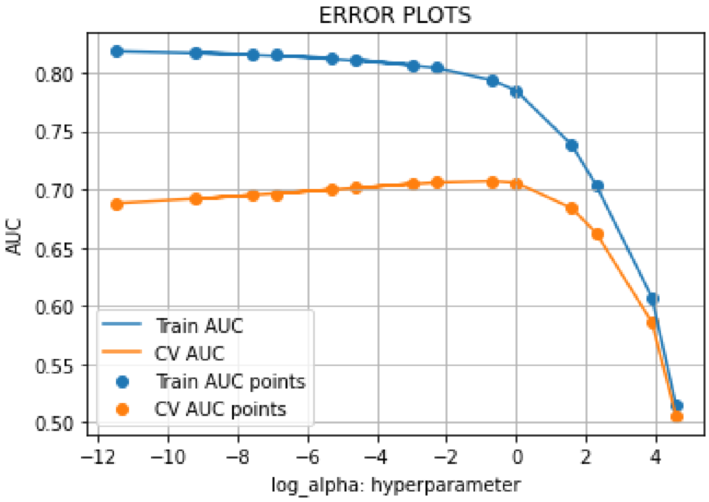
Apply NB on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

In [78]:

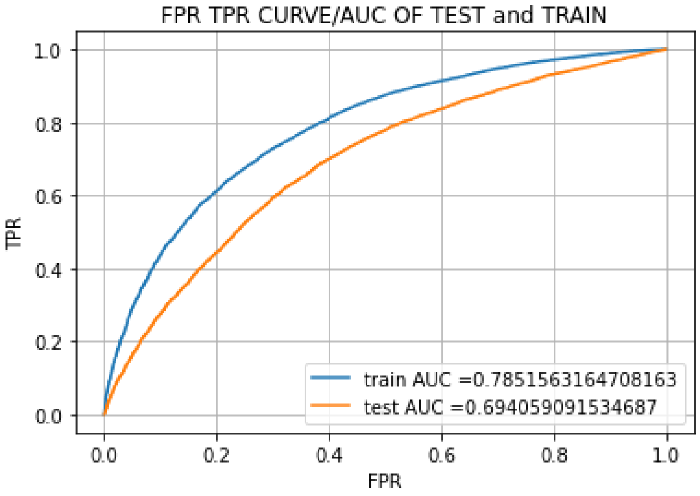
```
# please write all the code with proper documentation, and proper titles for each subsect
ion
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging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




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 = 49
000
    # 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


train_auc = []
cv_auc = []
log_alpha=[]
aplha = [0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
for i in tqdm(aplha):
    NB = MultinomialNB(alpha=i,class_prior= [0.5, 0.5])
    NB.fit(X_tr, y_train)
    y_train_pred = batch_predict(NB, X_tr)
    y_cv_pred = batch_predict(NB, X_cr)
    log_alpha.append(math.log(i))
    # 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(log_alpha, train_auc, label='Train AUC')
plt.plot(log_alpha, cv_auc, label='CV AUC')

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

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

#using best alpha =1
nb = MultinomialNB(alpha=1,class_prior= [0.5, 0.5])
nb.fit(X_tr, y_train)

#getting top positive and negative features
#https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-baye
s

def get_salient_words(nb_clf, vect, class_ind):
    """Return salient words for given class
    Parameters
    ----------
    nb_clf : a Naive Bayes classifier (e.g. MultinomialNB, BernoulliNB)
    vect : CountVectorizer
    class_ind : int
    Returns
    -------
    list
        a sorted list of (word, log prob) sorted by log probability in descending order.
    """

    words = vect.get_feature_names()
    zipped = list(zip(words, nb_clf.feature_log_prob_[class_ind]))
    sorted_zip = sorted(zipped, key=lambda t: t[1], reverse=True)

    return sorted_zip

neg_salient_top_20 = get_salient_words(nb, vectorizer, 0)[:20]
pos_salient_top_20 = get_salient_words(nb, vectorizer, 1)[:20]

print("top  20 features of positive and negative class")
print(neg_salient_top_20)
print(pos_salient_top_20)


y_train_pred = batch_predict(nb, X_tr)
y_test_pred = batch_predict(nb, X_te)



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("FPR")
plt.ylabel("TPR")
plt.title("FPR TPR CURVE/AUC OF TEST and TRAIN")
plt.grid()
plt.show()


print("="*100)
# 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

print("="*100)

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")
cm=confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
print(cm)
print("="*100)

#used below reference for confusion matrix heat map
#https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confus
ion-matrix-in-exponential-form-to-nor
ax= plt.subplot();
sns.heatmap(cm, annot=True,cmap='Blues',ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
```

```python
ax.set_ylim(2.0, 0)
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['0','1']);
ax.yaxis.set_ticklabels(['0','1']);
print("="*100)
```

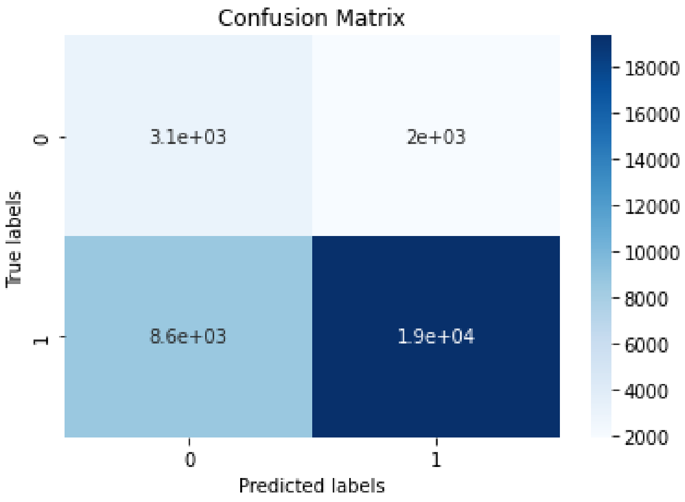
```
100%|██████████| 14/14 [00:03<00:00,  3.83it/s]
```



```
top  20 features of positive and negative class
[('civics_government', -7.741281846366859), ('charactereducation', -8.48228504810444), ('
health_lifescience', -8.75054903469912), ('specialneeds', -8.980123476343618), ('esl', -9
.456599620094972), ('care_hunger', -9.775053351213506), ('mathematics', -9.83011312839653
3), ('other', -9.950257440238598), ('literature_writing', -10.016215408030394), ('teamspo
rts', -10.468200531773451), ('health_wellness', -10.70936258859034), ('appliedsciences',
-10.755882604225233), ('college_careerprep', -10.755882604225233), ('environmentalscience
', -10.755882604225233), ('gym_fitness', -10.755882604225233), ('extracurricular', -11.09
2354840846445), ('parentinvolvement', -11.315498392160656), ('history_geography', -11.497
81994895461), ('visualarts', -11.72096350026882), ('literacy', -11.854494892893342)]
[('civics_government', -7.769127277727385), ('charactereducation', -8.35273549240603), ('
health_lifescience', -8.630897170723124), ('specialneeds', -9.029100082789503), ('esl', -
9.578278489667309), ('care_hunger', -9.723695489525813), ('other', -9.837140123583598), (
'literature_writing', -9.936028124735808), ('mathematics', -10.116201452991472), ('enviro
nmentalscience', -10.504644873530207), ('teamsports', -10.611412848955911), ('health_well
ness', -10.661965128118744), ('gym_fitness', -10.972120056422582), ('appliedsciences', -1
1.112892610303666), ('extracurricular', -11.347732201381067), ('college_careerprep', -11.
362547287166208), ('music', -11.783050272638912), ('parentinvolvement', -12.0408793819410
1), ('literacy', -12.070732345090693), ('history_geography', -12.166042524895017)]
```



```
====================================================================================
============
====================================================================================
============
the maximum value of tpr*(1-fpr) 0.5101257248671416 for threshold 0.511
Test confusion matrix
[[ 3054  1956]
 [ 8645 19345]]
====================================================================================
============
=============================================================================================
===========
```



SET2:TFIDF

Encoding using tfidf and applying NB

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

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

In [77]:

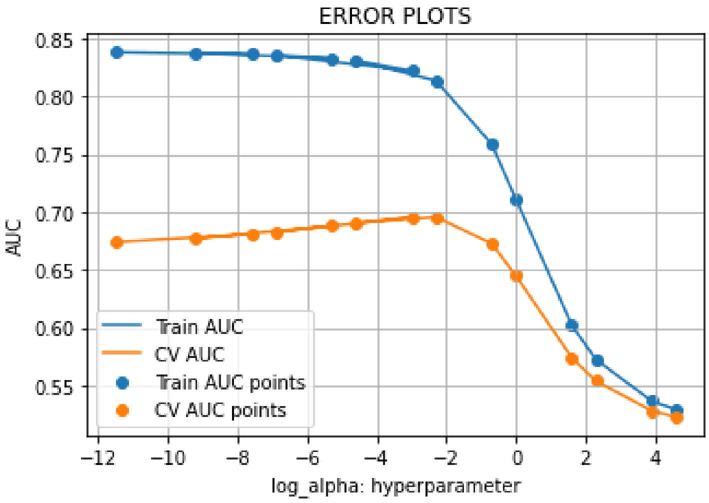
```
preprocessed_train_school_state = X_train['school_state'].values
preprocessed_test_school_state = X_test['school_state'].values


school_state_ohe_train_tfidf = vectorizer1.fit_transform(preprocessed_train_school_state)
school_state_ohe_cv_tfidf = vectorizer1.transform(X_cv['school_state'].values)
school_state_ohe_test_tfidf = vectorizer1.transform(preprocessed_test_school_state)
print("Shape of matrix after one hot encodig school_state_ohe_train",school_state_ohe_train.shape)
print("Shape of matrix after one hot encodig school_state_ohe_cv",school_state_ohe_cv.shape)
print("Shape of matrix after one hot encodig school_state_ohe_test",school_state_ohe_test.shape)


preprocessed_train_teacher_prefix = X_train['teacher_prefix'].values
preprocessed_test_teacher_prefix = X_test['teacher_prefix'].values

teacher_prefix_ohe_train_tfidf = vectorizer1.fit_transform(preprocessed_train_teacher_prefix)
teacher_prefix_ohe_cv_tfidf = vectorizer1.transform(X_cv['teacher_prefix'].values)
teacher_prefix_ohe_test_tfidf = vectorizer1.transform(preprocessed_test_teacher_prefix)
print("Shape of matrix after one hot encodig teacher_prefix_ohe_train_tfidf",teacher_prefix_ohe_train_tfidf.shape)
print("Shape of matrix after one hot encodig teacher_prefix_ohe_cv_tfidf",teacher_prefix_ohe_cv_tfidf.shape)
print("Shape of matrix after one hot encodig teacher_prefix_ohe_test_tfidf",teacher_prefix_ohe_test_tfidf.shape)



preprocessed_train_project_grade_category = X_train['project_grade_category'].values
preprocessed_test_project_grade_category = X_test['project_grade_category'].values
```

```
project_grade_category_ohe_train_tfidf = vectorizer1.fit_transform(preprocessed_train_project_grade_category)
project_grade_category_ohe_cv_tfidf = vectorizer1.transform(X_cv['project_grade_category'].values)
project_grade_category_ohe_test_tfidf = vectorizer1.transform(preprocessed_test_project_grade_category)
print("Shape of matrix after one hot encodig project_grade_category_ohe_train_tfidf",project_grade_category_ohe_train_tfidf.shape)
print("Shape of matrix after one hot encodig project_grade_category_ohe_cv_tfidf",project_grade_category_ohe_cv_tfidf.shape)
print("Shape of matrix after one hot encodig project_grade_category_ohe_test_tfidf",project_grade_category_ohe_test_tfidf.shape)

preprocessed_train_clean_categories = X_train['clean_categories'].values
preprocessed_test_clean_categories = X_test['clean_categories'].values


clean_categories_ohe_train_tfidf = vectorizer1.fit_transform(preprocessed_train_clean_categories)
clean_categories_ohe_cv_tfidf = vectorizer1.transform(X_cv['clean_categories'].values)
clean_categories_ohe_test_tfidf = vectorizer1.transform(preprocessed_test_clean_categories)
print("Shape of matrix after one hot encodig clean_categories_ohe_train_tfidf",clean_categories_ohe_train_tfidf.shape)
print("Shape of matrix after one hot encodig clean_categories_ohe_cv_tfidf",clean_categories_ohe_cv_tfidf.shape)
print("Shape of matrix after one hot encodig clean_categories_ohe_test_tfidf",clean_categories_ohe_test_tfidf.shape)

preprocessed_train_clean_subcategories = X_train['clean_subcategories'].values
preprocessed_test_clean_subcategories = X_test['clean_subcategories'].values

clean_subcategories_ohe_train_tfidf = vectorizer1.fit_transform(preprocessed_train_clean_subcategories)
clean_subcategories_ohe_cv_tfidf = vectorizer1.transform(X_cv['clean_subcategories'].values)
clean_subcategories_ohe_test_tfidf = vectorizer1.transform(preprocessed_test_clean_categories)
print("Shape of matrix after one hot encodig clean_subcategories_ohe_train_tfidf ",clean_subcategories_ohe_train_tfidf.shape)
print("Shape of matrix after one hot encodig clean_subcategories_ohe_cv_tfidf",clean_subcategories_ohe_cv_tfidf.shape)
print("Shape of matrix after one hot encodig  clean_subcategories_ohe_test_tfidf",clean_subcategories_ohe_test_tfidf.shape)




normalizer1 = Normalizer()
normalizer1.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm_tfidf = normalizer1.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_norm_tfidf= normalizer1.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm_tfidf= normalizer1.transform(X_test['price'].values.reshape(-1,1))

print("X_train_price_norm:",len(X_train_price_norm_tfidf))
print("X_cv_price_norm",len(X_cv_price_norm_tfidf))
print("X_test_price_norm",len(X_test_price_norm_tfidf))








#stacking using tfidf vectorizer
X_tr_tfidf= hstack((train_tfidf, school_state_ohe_train_tfidf, teacher_prefix_ohe_train_tfidf, project_grade_category_ohe_train_tfidf, X_train_price_norm_tfidf)).tocsr()
X_cr_tfidf = hstack((cv_tfidf, school_state_ohe_cv_tfidf, teacher_prefix_ohe_cv_tfidf, project_grade_category_ohe_cv_tfidf, X_cv_price_norm_tfidf)).tocsr()
X_te_tfidf = hstack((test_tfidf, school_state_ohe_test_tfidf, teacher_prefix_ohe_test_tf
idf, project_grade_category_ohe_test_tfidf, X_test_price_norm_tfidf)).tocsr()

print("Final Data matrix TFIDF")
print(X_tr_tfidf.shape, y_train.shape)
print(X_cr_tfidf.shape, y_cv.shape)
print(X_te_tfidf.shape, y_test.shape)
print("="*100)





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 = 49
000
    # 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

#calculating best apha using for loop
train_auc = []
cv_auc = []
log_alpha=[]
aplha = [0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
for i in tqdm(aplha):
    Naive_bayes = MultinomialNB(alpha=i,class_prior= [0.5, 0.5])
    Naive_bayes.fit(X_tr_tfidf, y_train)
    y_train_pred = batch_predict(Naive_bayes, X_tr_tfidf)
    y_cv_pred = batch_predict(Naive_bayes, X_cr_tfidf)
    log_alpha.append(math.log(i))
    # 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(log_alpha, train_auc, label='Train AUC')
plt.plot(log_alpha, cv_auc, label='CV AUC')

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

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

print("="*100)

#using best alpha =0.1 and training the model
nb_ = MultinomialNB(alpha=0.1,class_prior= [0.5, 0.5])
nb_.fit(X_tr_tfidf, y_train)

#getting top positive and negative features
#https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-baye
s

def get_salient_words(nb_clf, vect, class_ind):
    """Return salient words for given class
    Parameters
    ----------
    nb_clf : a Naive Bayes classifier (e.g. MultinomialNB, BernoulliNB)
    vect : CountVectorizer
    class_ind : int
    Returns
    -------
    list
        a sorted list of (word, log prob) sorted by log probability in descending order.
    """

    words = vect.get_feature_names()
    zipped = list(zip(words, nb_clf.feature_log_prob_[class_ind]))
    sorted_zip = sorted(zipped, key=lambda t: t[1], reverse=True)

    return sorted_zip

neg_salient_top_20 = get_salient_words(nb_, vectorizer1, 0)[:20]
pos_salient_top_20 = get_salient_words(nb_, vectorizer1, 1)[:20]

print("top  20 features of positive and negative class")
print(neg_salient_top_20)
print(pos_salient_top_20)


y_train_pred = batch_predict(nb_, X_tr_tfidf)     #predicting ytrain using best aplha
y_test_pred = batch_predict(nb_, X_te_tfidf)      #prediction y test  using best alpha



train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)#calculating FPR,T
PR train
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)#calculating FPR,TPR te
st

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("FPR")
plt.ylabel("TPR")
plt.title("FPR TPR CURVE/AUC OF TEST and TRAIN")
plt.grid()
plt.show()


print("="*100)
# 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

print("="*100)

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")
cm=confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

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

#used below reference for confusion matrix heat map
#https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confusion-matrix-in-exponential-form-to-nor
ax= plt.subplot();
sns.heatmap(cm, annot=True,cmap='Blues',ax=ax);
# labels, title and ticks
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_ylim(2.0, 0)
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['0','1']);
ax.yaxis.set_ticklabels(['0','1']);
print("="*100)
```

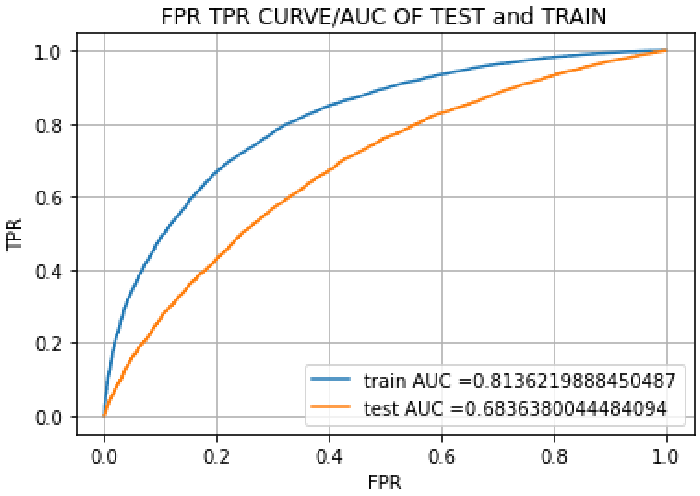
```
Shape of matrix after one hot encodig school_state_ohe_train (44890, 51)
Shape of matrix after one hot encodig school_state_ohe_cv (22110, 51)
Shape of matrix after one hot encodig school_state_ohe_test (33000, 51)
Shape of matrix after one hot encodig teacher_prefix_ohe_train_tfidf (44890, 4)
Shape of matrix after one hot encodig teacher_prefix_ohe_cv_tfidf (22110, 4)
Shape of matrix after one hot encodig teacher_prefix_ohe_test_tfidf (33000, 4)
Shape of matrix after one hot encodig project_grade_category_ohe_train_tfidf (44890, 4)
Shape of matrix after one hot encodig project_grade_category_ohe_cv_tfidf (22110, 4)
Shape of matrix after one hot encodig project_grade_category_ohe_test_tfidf (33000, 4)
Shape of matrix after one hot encodig clean_categories_ohe_train_tfidf (44890, 9)
Shape of matrix after one hot encodig clean_categories_ohe_cv_tfidf (22110, 9)
Shape of matrix after one hot encodig clean_categories_ohe_test_tfidf (33000, 9)
Shape of matrix after one hot encodig clean_subcategories_ohe_train_tfidf  (44890, 30)
Shape of matrix after one hot encodig clean_subcategories_ohe_cv_tfidf (22110, 30)
Shape of matrix after one hot encodig  clean_subcategories_ohe_test_tfidf (33000, 30)
X_train_price_norm: 44890
X_cv_price_norm 22110
X_test_price_norm 33000
Final Data matrix TFIDF
(44890, 11766) (44890,)
(22110, 11766) (22110,)
(33000, 11766) (33000,)
====================================================================================================
```

```
100%|██████████| 14/14 [00:03<00:00,  3.90it/s]
```

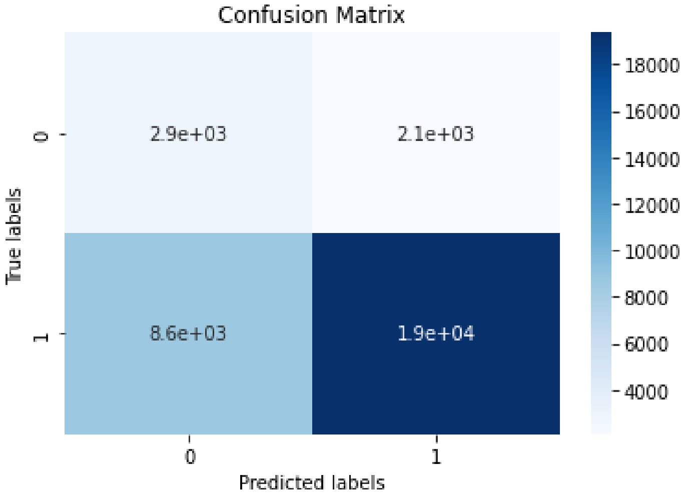


```
====================================================================================================
top  20 features of positive and negative class
[('civics_government', -7.958607733685349), ('charactereducation', -8.618695361332161), ('health_lifescience', -8.843307625671905), ('specialneeds', -8.979504493670795), ('esl', -9.396754032973295), ('care_hunger', -9.614525827088084), ('mathematics', -9.61487063703866), ('other', -9.85183189686178), ('literature_writing', -9.909336477744695), ('teamsports', -10.182789363410514), ('health_wellness', -10.408491800361649), ('gym_fitness', -10.472010605098385), ('college_careerprep', -10.487702050450528), ('appliedsciences', -10.494714333381145), ('environmentalscience', -10.499089935550618), ('extracurricular', -10.689554864335586), ('parentinvolvement', -10.910183254662735), ('history_geography', -11.032541571119792), ('visualarts', -11.213125291042545), ('literacy', -11.400486706965534)]
```

```
[('civics_government', -7.989904342905079), ('charactereducation', -8.485614538538055), (
'health_lifescience', -8.707024767261881), ('specialneeds', -9.050699835639088), ('esl',
-9.461000196386431), ('care_hunger', -9.603488500864495), ('other', -9.693069077137075),
('literature_writing', -9.804503480079056), ('mathematics', -9.898260081118565), ('enviro
nmentalscience', -10.30903873032398), ('teamsports', -10.381367592900165), ('health_welln
ess', -10.384070562234296), ('gym_fitness', -10.69638887502163), ('appliedsciences', -10.
798164765664199), ('college_careerprep', -10.977657961895972), ('extracurricular', -10.99
54659510422), ('music', -11.292747952765614), ('parentinvolvement', -11.595331984592033),
('literacy', -11.618637372994861), ('history_geography', -11.666610364657362)]
```



```
========================================================================================
===========
========================================================================================
===========
the maximum value of tpr*(1-fpr) 0.5438031966906057 for threshold 0.505
Test confusion matrix
[[ 2919  2091]
 [ 8634 19356]]
========================================================================================
===========
========================================================================================
===========
```



3. Summary

as mentioned in the step 5 of instructions

In [75]:

```
#https://www.geeksforgeeks.org/creating-tables-with-prettytable-library-python/
table=PrettyTable()
table.field_names =["vectorizer used","classification model","AUC SCORE OF TEST","best AL
PHA used"]
table.add_row(["BOW","NAIVE BAYES",0.694,1])
table.add_row(["TF-IDF","NAIVE BAYES",0.683,0.1])
print(table)
```

```
+-----------------+----------------------+-------------------+-----------------+
| vectorizer used | classification model | AUC SCORE OF TEST | best ALPHA used |
+-----------------+----------------------+-------------------+-----------------+
|       BOW       |     NAIVE BAYES      |       0.694       |        1        |
|      TF-IDF     |     NAIVE BAYES      |       0.683       |       0.1       |
+-----------------+----------------------+-------------------+-----------------+
```