Applying Multinomial Naive Bayes on DonorsChoose dataset

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
```

In [4]:
```python
# Loading the data
data = pd.read_csv('data.csv')
data.head()
```

Out[4]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_projects	project_is_approved	clean_categori
0	ca	mrs	grades_prek_2	53	1	math_scien
1	ut	ms	grades_3_5	4	1	specialnee
2	ca	mrs	grades_prek_2	10	1	literacy_langua
3	ga	mrs	grades_prek_2	2	1	appliedlearni
4	wa	mrs	grades_3_5	2	1	literacy_langua

In [8]:
```python
# Checking the columns and number of data points.
print(data.columns)
print(data.shape)
```

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

In [9]:
```python
# Droping the dependent varaible (project approval status) which needs to predicted.
y=data['project_is_approved'].values
X=data.drop(['project_is_approved'],axis=1)
```

In [10]:
```python
# Splitting the data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [11]:
```python
print('X Train:',X_train.shape)
print('X Test:',X_test.shape)
print('y Train:',y_train.shape)
print('X Test:',y_test.shape)
```

```
X Train: (87398, 8)
X Test: (21850, 8)
y Train: (87398,)
X Test: (21850,)
```

Bag of words approach

Encoding the categorical feature 'essay' with Bag of words to make data ready for the model.

In [12]:
```python
# Applying CountVectorizer (a.k.a. bag of words) to extend the features via one hot encoding.
features=[]
vectorizer = CountVectorizer(min_df=10)
vectorizer.fit(X_train['essay'].values)
X_train_essay_bow = vectorizer.transform(X_train['essay'].values)
X_test_essay_bow = vectorizer.transform(X_test['essay'].values)
f=vectorizer.get_feature_names()
features.extend(f)
print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_test_essay_bow.shape, y_test.shape)
```

```
After vectorizations
(87398, 15260) (87398,)
(21850, 15260) (21850,)
```

Encoding the other categorical featues with one hot encoding

In [13]:
```python
# Encoding feature school_state.
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values)
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
f=vectorizer.get_feature_names()
features.extend(f)
```

```
(87398, 51) (87398,)
(21850, 51) (21850,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv',
'wy']
```

In [14]:
```python
# Encoding feature teacher_prefix.
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values)
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_prefix'].values)
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
f=vectorizer.get_feature_names()
features.extend(f)
```

```
(87398, 5) (87398,)
(21850, 5) (21850,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
```

In [15]:
```python
# Encoding feature project_grade_category.
vectorizer = CountVectorizer()
vectorizer.fit(X_train['project_grade_category'].values)
X_train_grade_ohe = vectorizer.transform(X_train['project_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['project_grade_category'].values)
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
f=vectorizer.get_feature_names()
features.extend(f)
```

```
(87398, 4) (87398,)
(21850, 4) (21850,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
```

In [16]:
```python
# Encoding feature clean_categories.
vectorizer = CountVectorizer()#ngram_range=(1,2))
vectorizer.fit(X_train['clean_categories'].values)
X_train_category_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_test_category_ohe = vectorizer.transform(X_test['clean_categories'].values)
print(X_train_category_ohe.shape, y_train.shape)
print(X_test_category_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
f=vectorizer.get_feature_names()
features.extend(f)
```

```
(87398, 9) (87398,)
(21850, 9) (21850,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_scie
nce', 'music_arts', 'specialneeds', 'warmth']
```

In [17]:
```python
# Encoding feature clean_subcategories.
vectorizer = CountVectorizer(ngram_range=(1,1))
vectorizer.fit(X_train['clean_subcategories'].values)
#print(len(X_train['clean_subcategories'].unique()))
X_train_subcategory_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_test_subcategory_ohe = vectorizer.transform(X_test['clean_subcategories'].values)
print(X_train_subcategory_ohe.shape, y_train.shape)
print(X_test_subcategory_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
f=vectorizer.get_feature_names()
features.extend(f)
```

```
(87398, 30) (87398,)
(21850, 30) (21850,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_careerprep',
'communityservice', 'earlydevelopment', 'economics', 'environmentalscience', 'esl', 'extracurricula
r', 'financialliteracy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wellness',
'history_geography', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutritioneducation',
'other', 'parentinvolvement', 'performingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visu
alarts', 'warmth']
```

Normalizing the numerical features to get nuerical stability without losing information.

In [18]:
```python
# Normalizing feature price.
from sklearn.preprocessing import Normalizer
norm = Normalizer()
norm.fit(X_train['price'].values.reshape(1,-1))
X_train_price_norm = norm.transform(X_train['price'].values.reshape(1,-1))
X_test_price_norm = norm.transform(X_test['price'].values.reshape(1,-1))
X_train_price_norm = X_train_price_norm.reshape(-1,1)
X_test_price_norm = X_test_price_norm.reshape(-1,1)
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
features.append('price')
```

```
(87398, 1) (87398,)
(21850, 1) (21850,)
```

In [19]:
```python
# Normalizing feature teacher_number_of_previously_posted_projects.
norm.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_train_teacherpostedproject_norm = norm.transform(X_train['teacher_number_of_previously_posted_project
s'].values.reshape(1,-1))
X_test_teacherpostedproject_norm = norm.transform(X_test['teacher_number_of_previously_posted_projects'
].values.reshape(1,-1))
X_train_teacherpostedproject_norm=X_train_teacherpostedproject_norm.reshape(-1,1)
X_test_teacherpostedproject_norm=X_test_teacherpostedproject_norm.reshape(-1,1)
print(X_train_teacherpostedproject_norm.shape, y_train.shape)
print(X_test_teacherpostedproject_norm.shape, y_test.shape)
features.append('teacher_number_of_previously_posted_projects')
```

```
(87398, 1) (87398,)
(21850, 1) (21850,)
```

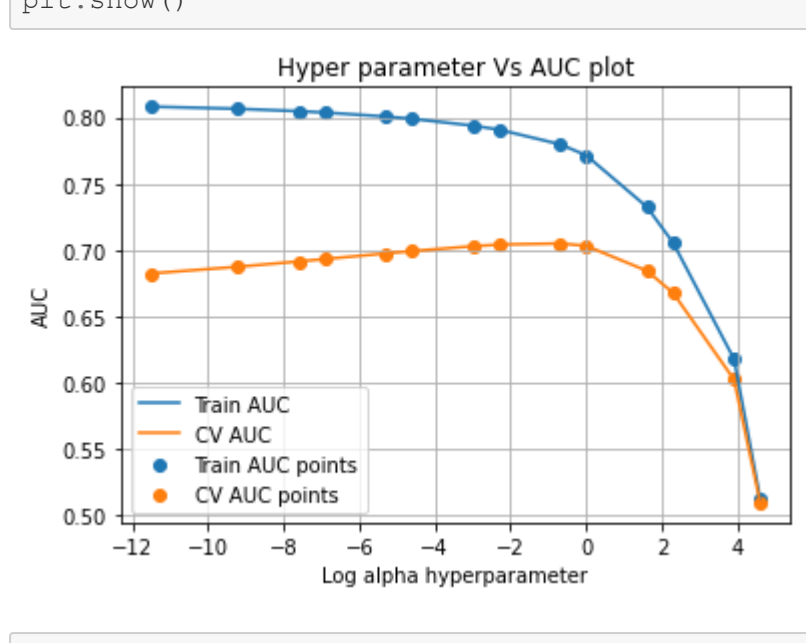
Combining all the above features into a final dataset in which our model will perform.

In [20]:
```python
# Using hstack from scipy to combine the features.
# https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.hstack.html
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_bow, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_ca
tegory_ohe, X_train_subcategory_ohe, X_train_price_norm, X_train_teacherpostedproject_norm)).tocsr()
X_te = hstack((X_test_essay_bow, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categor
y_ohe, X_test_subcategory_ohe, X_test_price_norm, X_test_teacherpostedproject_norm)).tocsr()

print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
```

```
(87398, 15361) (87398,)
(21850, 15361) (21850,)
```

Applying Naive Bayes on the dataset created above.

In [21]:
```python
# Prior to fit the model we need to find out the best value for our hyperparameter alpha (Laplace Smoot
hing)
# to avoid getting zero probability.
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import GridSearchCV

model=MultinomialNB()

# Assigning some values for our hyperparameter.
para={'alpha':[0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]}

# Using GridSearchCv to find the best value of our hyperparameter.
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
classifier=GridSearchCV(model, para, cv=5, scoring='roc_auc', return_train_score=True)
classifier.fit(X_tr, y_train)

# Putting the results of our cross-valication into a vairable.
results = pd.DataFrame.from_dict(classifier.cv_results_)
results = results.sort_values(['param_alpha'])

train_auc= results['mean_train_score']
cv_auc = results['mean_test_score']
alphas =  results['param_alpha']

# Coverting the alpahs to smaller values (using log) to better undertand the graph.
a_log=[]
for i in alphas:
    a_log.append(np.log(i))

# Plotting the graph of Train score, cross validation score and alpha.
plt.plot(a_log, train_auc, label='Train AUC')
plt.plot(a_log, cv_auc, label='CV AUC')

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


plt.legend()
plt.xlabel("Log alpha hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
```



In [23]:
```python
# Seeing our results of cross validation.
results.sort_values(['mean_test_score'],ascending=False)
```

Out[23]:

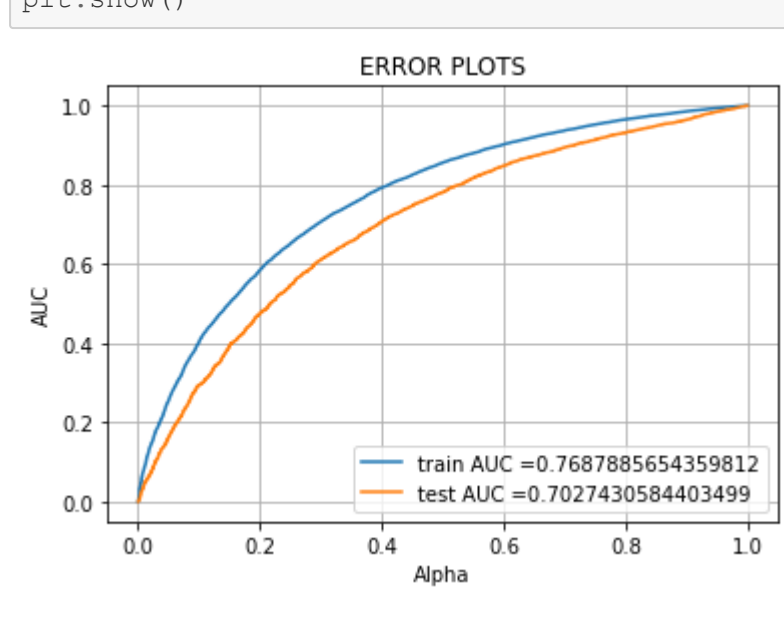
	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_score	split1_test_score	split2_test_sc
8	0.126429	0.007797	0.031245	0.000014	0.5	{'alpha': 0.5}	0.700466	0.699452	0.709
7	0.118877	0.007759	0.031251	0.000014	0.1	{'alpha': 0.1}	0.697697	0.698539	0.708
9	0.128821	0.010462	0.024048	0.010874	1	{'alpha': 1}	0.700090	0.697633	0.707
5	0.134429	0.007697	0.021866	0.007660	0.05	{'alpha': 0.05}	0.695913	0.697210	0.707
6	0.124997	0.009876	0.025003	0.007656	0.01	{'alpha': 0.01}	0.691462	0.693289	0.704
3	0.126442	0.010301	0.025006	0.007659	0.005	{'alpha': 0.005}	0.689551	0.691430	0.702
4	0.137483	0.006246	0.015625	0.000015	0.001	{'alpha': 0.001}	0.685284	0.687112	0.698
1	0.128130	0.006251	0.024995	0.007658	0.0005	{'alpha': 0.0005}	0.683512	0.685344	0.696
2	0.137487	0.015304	0.024995	0.007658	0.0001	{'alpha': 0.0001}	0.679748	0.681480	0.691
10	0.136997	0.010926	0.019181	0.006090	5	{'alpha': 5}	0.685882	0.679202	0.687
0	0.131924	0.021787	0.030399	0.010027	1e-05	{'alpha': 1e-05}	0.675058	0.676705	0.686
11	0.125004	0.009888	0.034362	0.006246	10	{'alpha': 10}	0.670926	0.662981	0.669
12	0.121868	0.006250	0.024998	0.007654	50	{'alpha': 50}	0.605545	0.599809	0.598
13	0.127586	0.006587	0.018759	0.006252	100	{'alpha': 100}	0.510065	0.511814	0.509

14 rows × 21 columns

From the results above we can see that the best CV score comes out when hyperparameter alpha=0.5

In [28]:
```python
# Function to get output probabilities on the data not the predicted outputs.
def batch_predict(clf, data):
    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000

    # In this for loop we will iterate until the last 1000 multiplier.
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])

    # This if condition for remaining data points
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [29]:
```python
# Initializing the model with best value of alpha.
from sklearn.metrics import roc_curve, auc

model=MultinomialNB(alpha=0.5)
model.fit(X_tr,y_train)

# Getting the prediction values from the above function.
train_predictions=batch_predict(model, X_tr)
test_predictions=batch_predict(model, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, train_predictions)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, test_predictions)

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("Alpha")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



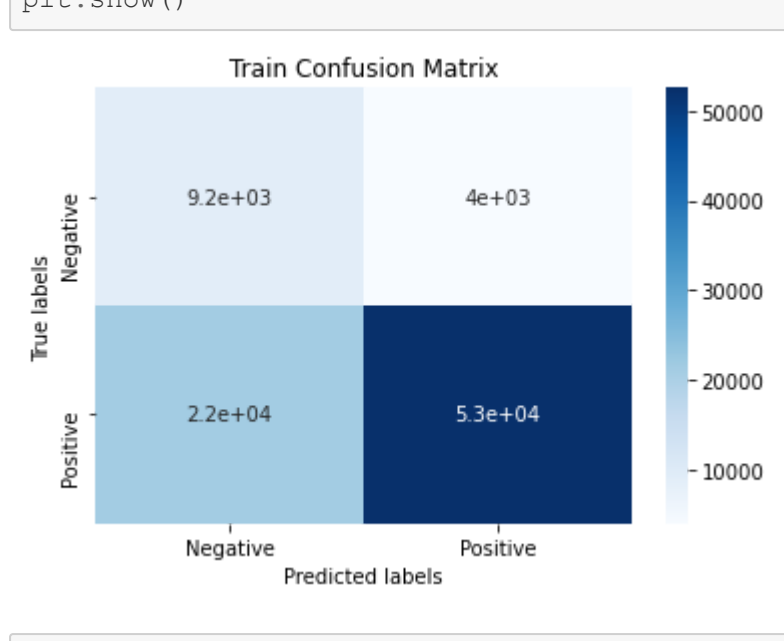
Train and Test AUC and score are 0.768 and 0.702 is achieved respectively.

In [30]:
```python
# Finding the best thershold for our model.
def find_best_threshold(threshold, fpr, tpr):
    t = threshold[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

# Making predictions with best threshold
def predict_with_best_t(proba, threshold):
    predictions = []
    for i in proba:
        if i>=threshold:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

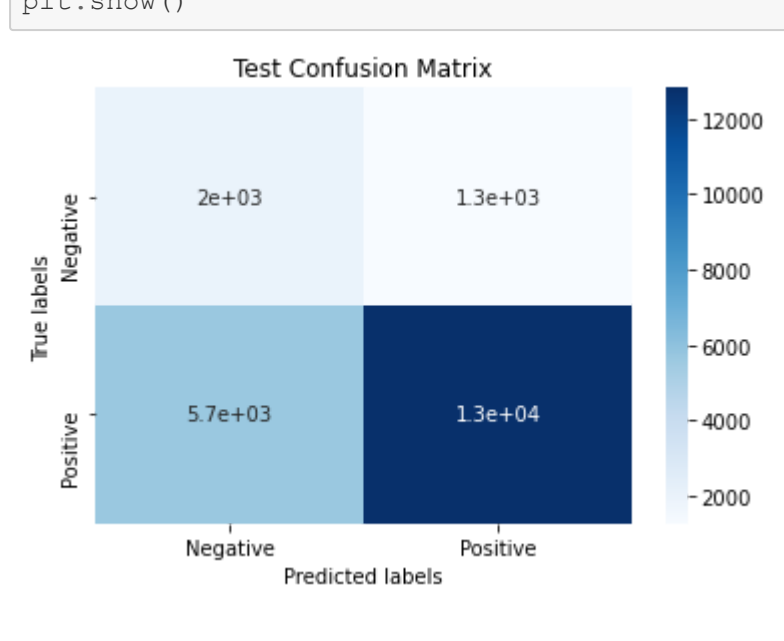
In [31]:
```python
# Getting the predictions from best threshold and printing the confusion matrix.
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
train_conf=confusion_matrix(y_train, predict_with_best_t(train_predictions, best_t))
print(train_conf)
print("Test confusion matrix")
test_conf=confusion_matrix(y_test, predict_with_best_t(test_predictions, best_t))
print(test_conf)
```

```
the maximum value of tpr*(1-fpr) 0.49513007707811524 for threshold 0.83
Train confusion matrix
[[ 9245  3989]
 [21599 52565]]
Test confusion matrix
[[ 2036  1272]
 [ 5720 12822]]
```

In [34]:
```python
# Ploting the confusion matrix heatmap for training data.
# https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confusion-matrix-in
-exponential-form-to-nor
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot();
sns.heatmap(train_conf, 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('Train Confusion Matrix');
ax.xaxis.set_ticklabels(['Negative','Positive']);
ax.yaxis.set_ticklabels(['Negative','Positive']);
plt.show()
```



In [35]:
```python
# Ploting the confusion matrix heatmap for training data.
#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(test_conf, 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('Test Confusion Matrix');
ax.xaxis.set_ticklabels(['Negative','Positive']);
ax.yaxis.set_ticklabels(['Negative','Positive']);
plt.show()
```



TFIDF approach

Repeating all the above steps but replacing the Bag of words approach for feature 'essay' to TFIDF approach.

In [36]:
```python
data = pd.read_csv('data.csv')
#data.head()
y=data['project_is_approved'].values
X=data.drop(['project_is_approved'],axis=1)
```

In [37]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)
```

In [38]:
```python
print('X Train:',X_train.shape)
print('X Test:',X_test.shape)
print('y Train:',y_train.shape)
print('X Test:',y_test.shape)
```

```
X Train: (87398, 8)
X Test: (21850, 8)
y Train: (87398,)
X Test: (21850,)
```

Encoding the categorical feature 'essay' with TFIDF vectorizer to make data ready for the model.

In [39]:
```python
# Using scikit-learn TFIDF vectorizer.
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['essay'].values)
X_train_essay_tfidf = vectorizer.transform(X_train['essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
```

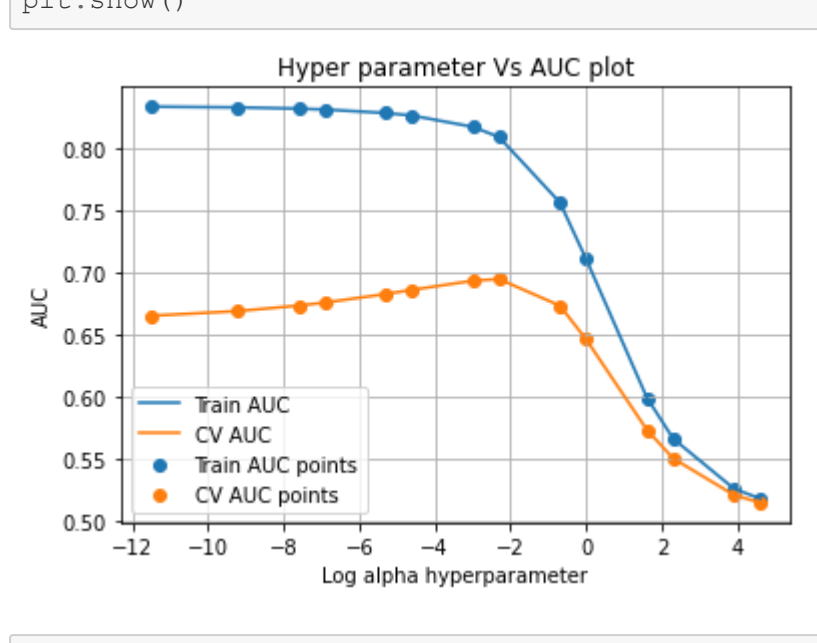
```
After vectorizations
(87398, 15268) (87398,)
(21850, 15268) (21850,)
```

In [40]:
```python
# Combining all other features with tfidf's output.
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_tfidf, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_
category_ohe, X_train_subcategory_ohe, X_train_price_norm, X_train_teacherpostedproject_norm)).tocsr()
X_te = hstack((X_test_essay_tfidf, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categ
ory_ohe, X_test_subcategory_ohe, X_test_price_norm, X_test_teacherpostedproject_norm)).tocsr()

print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
```

```
(87398, 15369) (87398,)
(21850, 15369) (21850,)
```

Selecting best hyperparameter as done above.

In [44]:
```python
model=MultinomialNB()
para={'alpha':[0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]}
classifier=GridSearchCV(model, para, cv=5, scoring='roc_auc', return_train_score=True)
classifier.fit(X_tr, y_train)

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

train_auc= results['mean_train_score']
cv_auc = results['mean_test_score']
alphas =  results['param_alpha']

a_log=[]
for i in alphas:
    a_log.append(np.log(i))


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

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


plt.legend()
plt.xlabel("Log alpha hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
```



In [45]:
```python
results.sort_values(['mean_test_score'],ascending=False)
```

Out[45]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_alpha	params	split0_test_score	split1_test_score	split2_test_sc
7	0.131240	0.007654	0.021876	0.007647	0.1	{'alpha': 0.1}	0.690178	0.695416	0.701
5	0.134355	0.012497	0.018752	0.006257	0.05	{'alpha': 0.05}	0.688609	0.693080	0.700
6	0.121869	0.006253	0.024998	0.007664	0.01	{'alpha': 0.01}	0.680702	0.684085	0.694
3	0.130168	0.011607	0.025354	0.007980	0.005	{'alpha': 0.005}	0.677126	0.680233	0.691
4	0.141271	0.013693	0.024275	0.007295	0.001	{'alpha': 0.001}	0.669870	0.672427	0.685
1	0.131237	0.015935	0.021881	0.007654	0.0005	{'alpha': 0.0005}	0.667362	0.669724	0.683
8	0.134374	0.012497	0.021868	0.007654	0.5	{'alpha': 0.5}	0.669926	0.677920	0.674
2	0.133040	0.012748	0.024878	0.007564	0.0001	{'alpha': 0.0001}	0.663017	0.664884	0.679
0	0.122899	0.013426	0.028122	0.006259	1e-05	{'alpha': 1e-05}	0.659437	0.660822	0.676
9	0.121871	0.011693	0.025011	0.007656	1	{'alpha': 1}	0.643929	0.652438	0.644
10	0.128113	0.006248	0.018752	0.006250	5	{'alpha': 5}	0.571699	0.578200	0.569
11	0.131238	0.007653	0.021873	0.007658	10	{'alpha': 10}	0.547746	0.553829	0.546
12	0.128106	0.011695	0.021871	0.007650	50	{'alpha': 50}	0.516421	0.521145	0.517
13	0.123545	0.002912	0.028131	0.006246	100	{'alpha': 100}	0.510382	0.514093	0.511

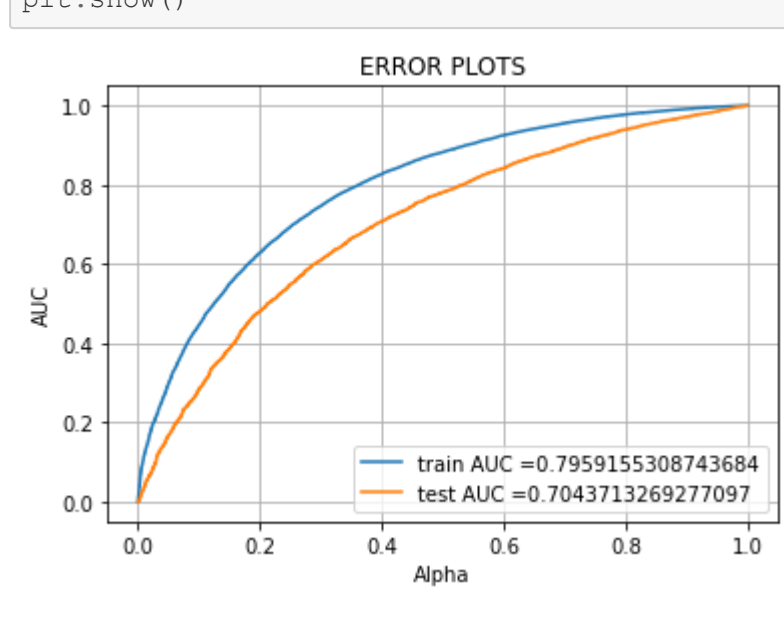
14 rows × 21 columns

From the results above we can see that the best CV score comes out when hyperparameter alpha=0.1

Getting probabilities

In [46]:
```python
def batch_predict(clf, data):

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000

    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])

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

    return y_data_pred
```

In [47]:
```python
from sklearn.metrics import roc_curve, auc

model=MultinomialNB(alpha=0.1)
model.fit(X_tr,y_train)
train_predictions=batch_predict(model, X_tr)
test_predictions=batch_predict(model, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, train_predictions)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, test_predictions)

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("Alpha")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Predicting with best threshold.

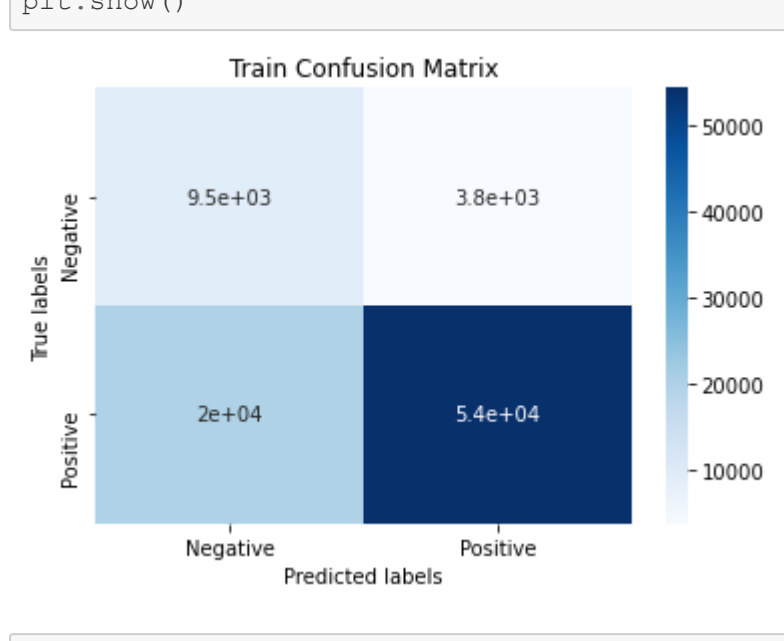
In [48]:
```python
def find_best_threshold(threshold, fpr, tpr):
    t = threshold[np.argmax(tpr*(1-fpr))]
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

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

Confusion Matrix

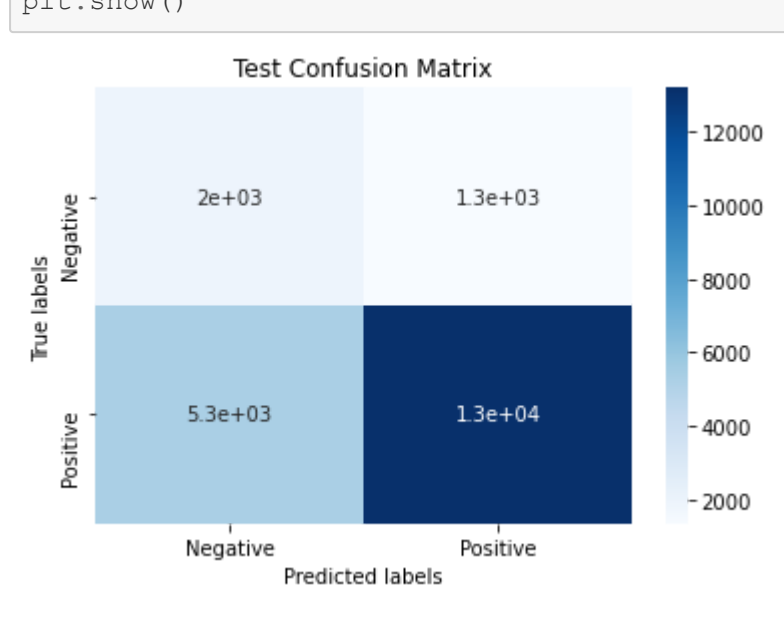
In [49]:
```python
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
train_conf=confusion_matrix(y_train, predict_with_best_t(train_predictions, best_t))
print(train_conf)
print("Test confusion matrix")
test_conf=confusion_matrix(y_test, predict_with_best_t(test_predictions, best_t))
print(test_conf)
```

```
the maximum value of tpr*(1-fpr) 0.5239064877248993 for threshold 0.846
Train confusion matrix
[[ 9468  3766]
 [19854 54310]]
Test confusion matrix
[[ 1966  1342]
 [ 5340 13202]]
```

In [50]:
```python
#https://stackoverflow.com/questions/61748441/how-to-fix-the-values-displayed-in-a-confusion-matrix-in-
exponential-form-to-nor
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot();
sns.heatmap(train_conf, 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('Train Confusion Matrix');
ax.xaxis.set_ticklabels(['Negative','Positive']);
ax.yaxis.set_ticklabels(['Negative','Positive']);
plt.show()
```



In [51]:
```python
#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(test_conf, 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('Test Confusion Matrix');
ax.xaxis.set_ticklabels(['Negative','Positive']);
ax.yaxis.set_ticklabels(['Negative','Positive']);
plt.show()
```



Summary

Vectorizer	Hyperparameter	Test AUC
BOW	0.5	0.70
TFIDF	0.1	0.70