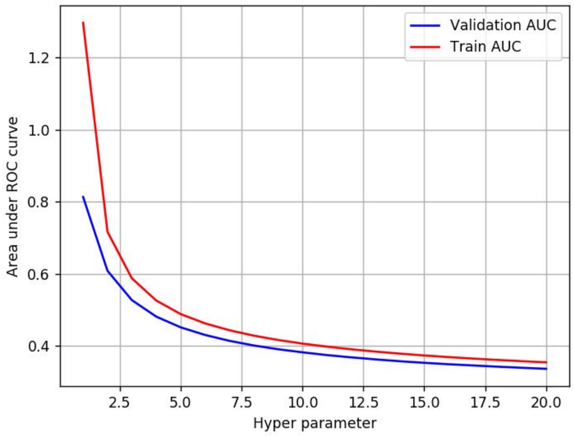
Assignment 6: Apply NB

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

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

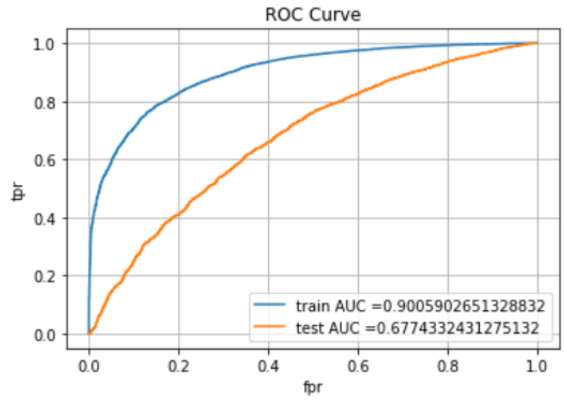
   - Set 1: categorical, numerical features + preprocessed_eassay (BOW)
   - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)
6. **The hyper paramter tuning(find best alpha:smoothing parameter)**

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



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

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



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

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

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

7. find the top 20 features from either from feature Set 1 or feature Set 2 using values of `feature_log_prob_ ` parameter of `MultinomialNB` (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print **BOTH** positive as well as negative corresponding feature names.
   - go through the link (https://imgur.com/mWvE7gj)
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

2. Naive Bayes

1.1 Loading Data

In [1]:

```
!pip install chart_studio
```

```
Collecting chart_studio
  Downloading https://files.pythonhosted.org/packages/ca/ce/330794a6
b6ca4b9182c38fc69dd2a9cbff60fd49421cb8648ee5fee352dc/chart_studio-1.
1.0-py3-none-any.whl (64kB)
     |████████████████████████████████| 71kB 6.1MB/s
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-
packages (from chart_studio) (1.15.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/
dist-packages (from chart_studio) (2.23.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/pyt
hon3.7/dist-packages (from chart_studio) (1.3.3)
Requirement already satisfied: plotly in /usr/local/lib/python3.7/di
st-packages (from chart_studio) (4.4.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python
3.7/dist-packages (from requests->chart_studio) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/p
ython3.7/dist-packages (from requests->chart_studio) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.2
1.1 in /usr/local/lib/python3.7/dist-packages (from requests->chart_
studio) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/
python3.7/dist-packages (from requests->chart_studio) (2020.12.5)
Installing collected packages: chart-studio
Successfully installed chart-studio-1.1.0
```

In [2]:

```
%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
import pickle
from tqdm import tqdm
import os
import chart_studio.plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

In [3]:

```
# mount g-drive
from google.colab import drive
drive.mount('/content/gdrive')
```

Mounted at /content/gdrive

In [4]:

```
# read data from csv into pandas dataframe data
import pandas
data = pandas.read_csv('/content/gdrive/MyDrive/6_Donors_choose_NB/preprocessed_
data.csv', nrows=50000)
```

In [5]:

```
# separate inputs (X) from output variable (y)
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)
X.head(1)
```

Out[5]:

	school_state	teacher_prefix	project_grade_category	teacher_number_of_previously_posted_p
0	ca	mrs	grades_prek_2	

In [6]:

```
X.shape
```

Out[6]:

```
(50000, 8)
```

In [7]:

```
y.shape
```

Out[7]:

```
(50000,)
```

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

In [8]:

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

1.3 Make Data Model Ready: encoding essay

In [9]:

```
# Convert a collection of text documents to a matrix of token counts
print("Before Vectorization:")
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vectorizer_essay = CountVectorizer(min_df=10, ngram_range=(1,4), max_features=50
00)
vectorizer_essay.fit(X_train['essay'].values) # fit has to happen only on train
 data

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

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```

```
Before Vectorization:
(22445, 8) (22445,)
(11055, 8) (11055,)
(16500, 8) (16500,)
====================================================================
================================
After vectorizations
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
====================================================================
================================
```

1.4 Make Data Model Ready: encoding numerical, categorical features

In [10]:

```
#Encodng Categorical feature: school_state
vectorizer_state = CountVectorizer()
vectorizer_state.fit(X_train['school_state'].values) # fit has to happen only on
train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer_state.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer_state.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer_state.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer_state.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga',
'hi', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'm
i', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'n
v', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'u
t', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
======================================================================
==============================
```

In [11]:

```
#Encodng Categorical feature: teacher_prefix
vectorizer_prefix = CountVectorizer()
vectorizer_prefix.fit(X_train['teacher_prefix'].values) # fit has to happen only
on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer_prefix.transform(X_train['teacher_prefix'].valu
es)
X_cv_teacher_ohe = vectorizer_prefix.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer_prefix.transform(X_test['teacher_prefix'].values
)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer_prefix.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 5) (22445,)
(11055, 5) (11055,)
(16500, 5) (16500,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
======================================================================
==============================
```

In [12]:

```
# Encoding Categorical Feature: project_grade_category
vectorizer_grade = CountVectorizer()
vectorizer_grade.fit(X_train['project_grade_category'].values) # fit has to happ
en only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer_grade.transform(X_train['project_grade_category']
.values)
X_cv_grade_ohe = vectorizer_grade.transform(X_cv['project_grade_category'].value
s)
X_test_grade_ohe = vectorizer_grade.transform(X_test['project_grade_category'].v
alues)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer_grade.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
======================================================================
================================
```

In [13]:

```
# Ecncoding Categorical Feature: clean_categories
vectorizer_clean = CountVectorizer()
vectorizer_clean.fit(X_train['clean_categories'].values) # fit has to happen onl
y on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_ohe = vectorizer_clean.transform(X_train['clean_categories'].value
s)
X_cv_clean_ohe = vectorizer_clean.transform(X_cv['clean_categories'].values)
X_test_clean_ohe = vectorizer_clean.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_clean_ohe.shape, y_train.shape)
print(X_cv_clean_ohe.shape, y_cv.shape)
print(X_test_clean_ohe.shape, y_test.shape)
print(vectorizer_clean.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 7) (22445,)
(11055, 7) (11055,)
(16500, 7) (16500,)
['appliedlearning', 'health_sports', 'history_civics', 'literacy_lan
guage', 'math_science', 'music_arts', 'specialneeds']
======================================================================
================================
```

In [14]:

```
# Ecncoding Categorical Feature: clean_subcategories
vectorizer_cleansub = CountVectorizer()
vectorizer_cleansub.fit(X_train['clean_subcategories'].values) # fit has to happ
en only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleansub_ohe = vectorizer_cleansub.transform(X_train['clean_subcategorie
s'].values)
X_cv_cleansub_ohe = vectorizer_cleansub.transform(X_cv['clean_subcategories'].va
lues)
X_test_cleansub_ohe = vectorizer_cleansub.transform(X_test['clean_subcategories'
].values)

print("After vectorizations")
print(X_train_cleansub_ohe.shape, y_train.shape)
print(X_cv_cleansub_ohe.shape, y_cv.shape)
print(X_test_cleansub_ohe.shape, y_test.shape)
print(vectorizer_cleansub.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 28) (22445,)
(11055, 28) (11055,)
(16500, 28) (16500,)
['appliedsciences', 'charactereducation', 'civics_government', 'coll
ege_careerprep', 'communityservice', 'earlydevelopment', 'economic
s', 'environmentalscience', 'esl', 'extracurricular', 'financiallite
racy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'hea
lth_wellness', 'history_geography', 'literacy', 'literature_writin
g', 'mathematics', 'music', 'nutritioneducation', 'other', 'parentin
volvement', 'performingarts', 'socialsciences', 'specialneeds', 'tea
msports', 'visualarts']
====================================================================
================================
```

In [16]:

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

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

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

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================
================================
```

In [17]:

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

X_train_teacherno_norm = normalizer_teacher.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_cv_teacherno_norm = normalizer_teacher.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_teacherno_norm = normalizer_teacher.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_teacherno_norm.shape, y_train.shape)
print(X_cv_teacherno_norm.shape, y_cv.shape)
print(X_test_teacherno_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

Concatinating all the features

In [19]:

```
# merge sparse matrices of encoded features: https://stackoverflow.com/a/1971064
8/4084039
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_bow, X_train_state_ohe, X_train_teacher_ohe, X_trai
n_grade_ohe, X_train_clean_ohe, X_train_cleansub_ohe, X_train_price_norm, X_trai
n_teacherno_norm)).tocsr()
X_cr = hstack((X_cv_essay_bow, X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe,
X_cv_clean_ohe, X_cv_cleansub_ohe, X_cv_price_norm, X_cv_teacherno_norm)).tocsr
()
X_te = hstack((X_test_essay_bow, X_test_state_ohe, X_test_teacher_ohe, X_test_gr
ade_ohe, X_test_clean_ohe, X_test_cleansub_ohe, X_test_price_norm, X_test_teache
rno_norm)).tocsr()

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

```
Final Data matrix
(22445, 5097) (22445,)
(11055, 5097) (11055,)
(16500, 5097) (16500,)
======================================================================
================================
```

1.5 Appling NB on dataset1: BoW

In [20]:

```
# function to return the probability score
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates 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 cr_loop will be 49041 - 49041%
1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [21]:

```
# The roc_auc_score function computes the area under the receiver operating char
acteristic (ROC) curve,
# By computing the area under the roc curve, the curve information is summarized
in one number.
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_alphas = []

alphas = [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(alphas):
    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)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates 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))

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)
```

```
100%|██████████| 14/14 [00:01<00:00,  7.48it/s]
100%|██████████| 14/14 [00:00<00:00, 71697.50it/s]
```

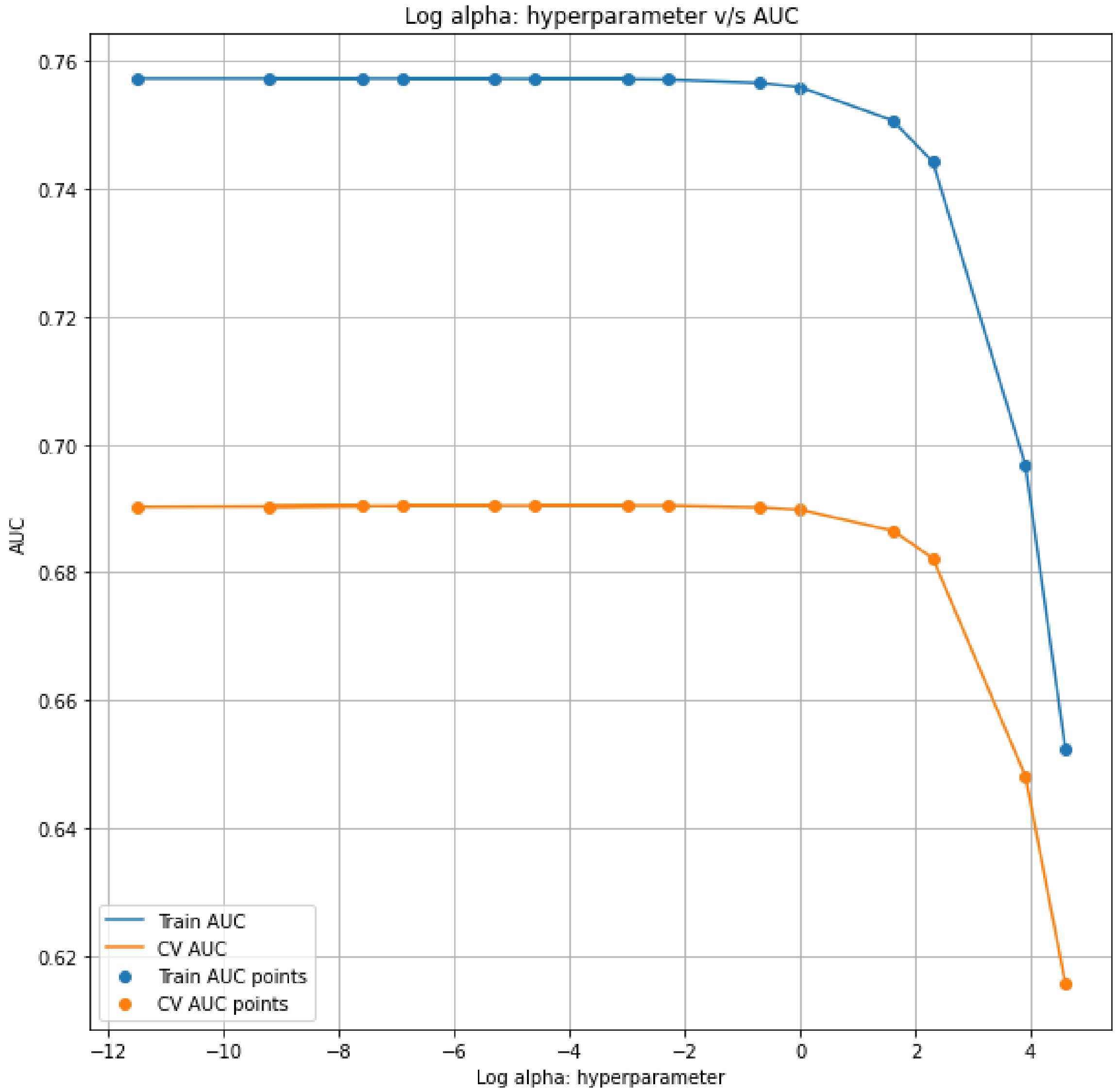
In [22]:

```
print (train_auc)
print (cv_auc)
```

```
[0.7572266889984984, 0.7572212142564643, 0.7572243574372008, 0.75721
02721507545, 0.7572197090713229, 0.7571452835687644, 0.7572028495220
668, 0.7570792620165792, 0.7565506026274335, 0.7558906527264879, 0.7
506714306268285, 0.7442092797615315, 0.6967583403120308, 0.652353408
6171849]
[0.6902231288543272, 0.6903371880409749, 0.6902893312523388, 0.69040
25994176866, 0.6903592757895762, 0.6904407814073487, 0.6904226183413
5, 0.6904180243330321, 0.6901313095357617, 0.6897546921253829, 0.686
490664427468, 0.6822001344736209, 0.6480682955620662, 0.615616829282
3916]
```

In [23]:

```
# plot the figure of log alpha (hyper-paramater) and AUC
plt.figure(figsize=(10, 10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("Log alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



We have started with hyperparameter alpha with as low as 0.00001 to 1000. Since it is difficult to plot the given range we have used log alphas on x-axis and Auc on y axis as shown in the plot.

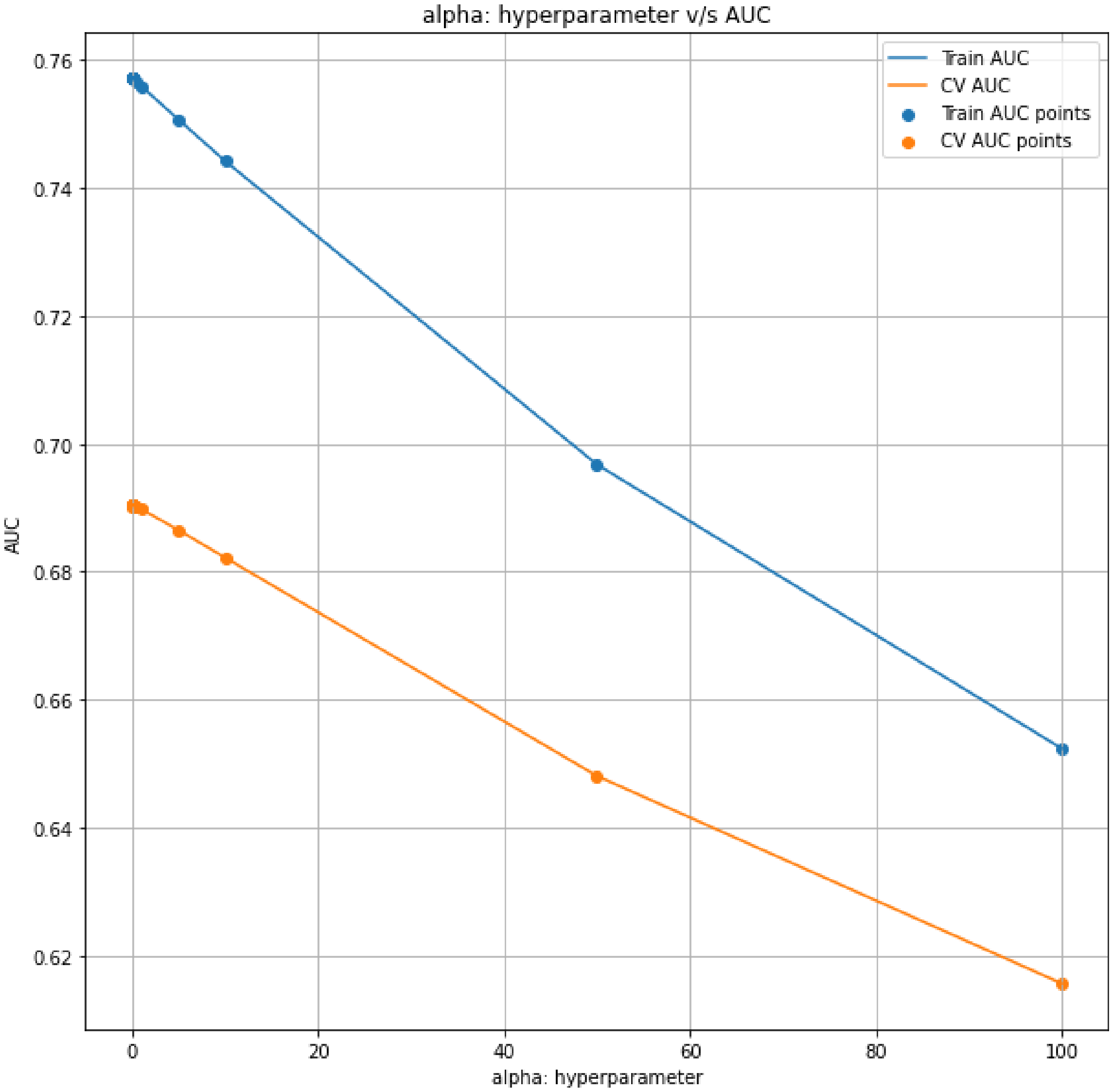
One of the main reason for using log scale is log scales allow a large range to be displayed without small values being compressed down into bottom of the graph.

we observe that as log alpha approaches close to 4, both train AUC and CV AUC lines converge

Using this plot we see after alpha=10 both lines converge at a much higher rate

In [24]:

```
# just to see how graph between alpha and AUC looks like
plt.figure(figsize=(10, 10))
plt.plot(alphas, train_auc, label='Train AUC')
plt.plot(alphas, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```


alpha: hyperparameter v/s AUC
Train AUC
CV AUC
Train AUC points
CV AUC points
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
AUC
0
20
40
60
80
100
alpha: hyperparameter

Here getting highest AUC when aplha = .0001

Gridsearch-cv using cv = 10 (K fold cross validation)

In [25]:

```
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB(class_prior=[0.5,0.5])
parameters = {'alpha': [0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1,
0.5, 1, 5, 10, 50, 100]}
clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc',return_train_score=
True,verbose=2)

clf.fit(X_tr, y_train)
train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
Fitting 10 folds for each of 14 candidates, totalling 140 fits
[CV] alpha=1e-05
....................................................
[CV] ..................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] ..................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] ..................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurr
ent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaini
ng:    0.0s
```

```
[CV] ....................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05 
......................................................
[CV] ....................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05 
......................................................
[CV] ....................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05 
......................................................
[CV] ....................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05 
......................................................
[CV] ....................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05 
......................................................
[CV] ....................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05 
......................................................
[CV] ....................................... alpha=1e-05, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005 
.....................................................
[CV] ...................................... alpha=0.0005, total=   0.
```

```
0s
[CV] alpha=0.0005 .....................................................
[CV] ...................................... alpha=0.0005, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.0s
```

```
[CV] alpha=0.005 
.....................................................
[CV] ...................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 
.....................................................
[CV] ...................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 
.....................................................
[CV] ...................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 
.....................................................
[CV] ...................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 
.....................................................
[CV] ...................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 
.....................................................
[CV] ...................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001 
```

```
.....................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
```

```
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
```

```
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=0.5 
........................................................
[CV] ........................................ alpha=0.5, total=   0.
0s
[CV] alpha=1 
..........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1 
..........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1 
..........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1 
..........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1 
..........................................................
[CV] .......................................... alpha=1, total=   0.
0s
```

```
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=10
```

```
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
0s
[CV] alpha=10
.......................................................
[CV] ........................................ alpha=10, total=   0.
1s
[CV] alpha=50
.......................................................
[CV] ........................................ alpha=50, total=   0.
0s
[CV] alpha=50
.......................................................
[CV] ........................................ alpha=50, total=   0.
0s
[CV] alpha=50
.......................................................
[CV] ........................................ alpha=50, total=   0.
0s
[CV] alpha=50
.......................................................
[CV] ........................................ alpha=50, total=   0.
0s
[CV] alpha=50
.......................................................
[CV] ........................................ alpha=50, total=   0.
0s
[CV] alpha=50
.......................................................
```

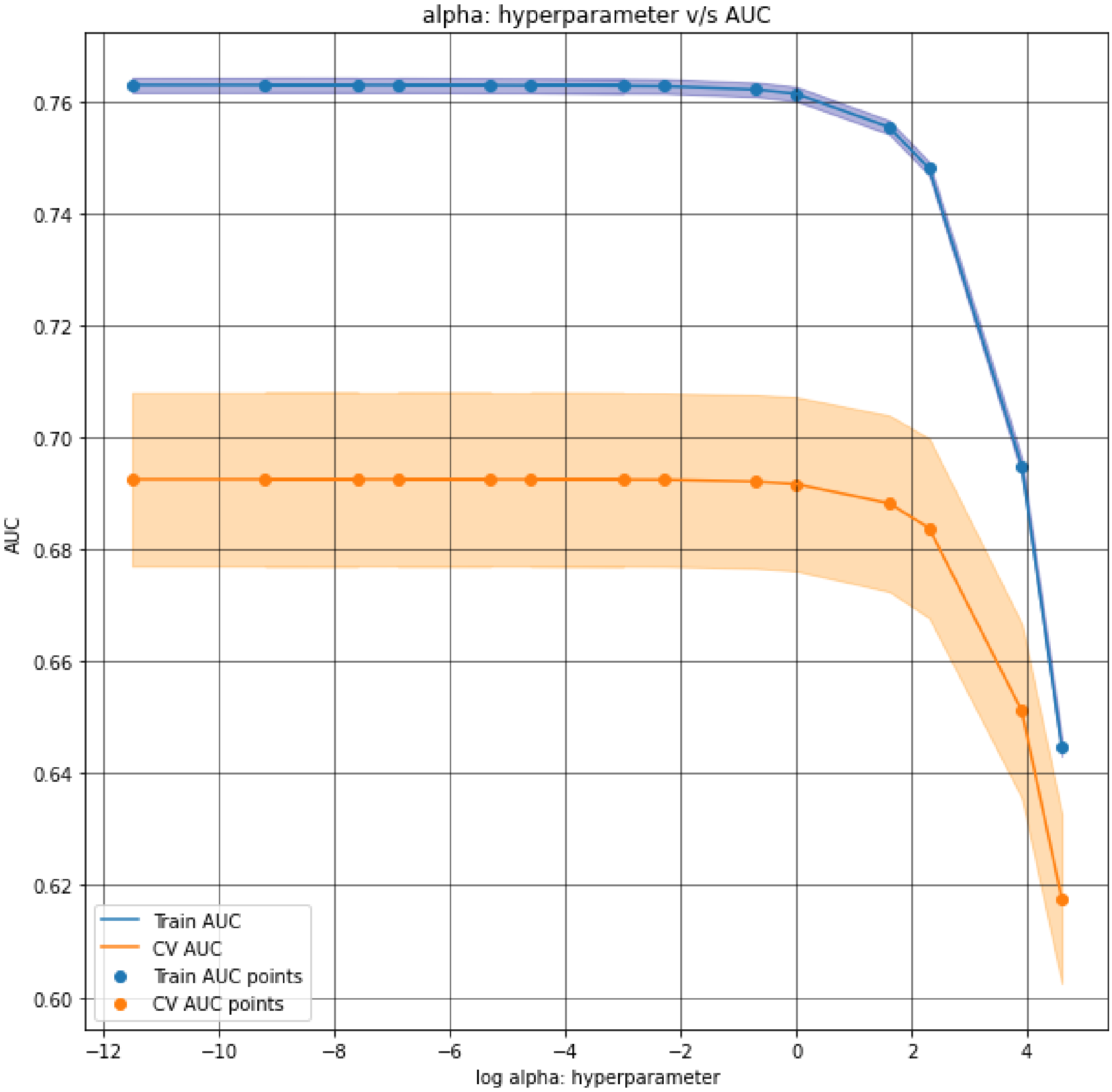
```
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
[CV] alpha=100
.......................................................
[CV] ........................................ alpha=100, total=   0.
0s
```

```
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    9.3s finishe
d
```

In [26]:

```
# plot the graph between log alpha and AUC
alphas = [0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1, 0.5, 1, 5, 10,
50, 100]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(10,10))

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas, train_auc - train_auc_std, train_auc + train_
auc_std, alpha=0.3, color='darkblue')

plt.plot(log_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas, cv_auc - cv_auc_std, cv_auc + cv_auc_std, alp
ha=0.3, color='darkorange')

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


plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
100%|██████████| 14/14 [00:00<00:00, 4568.60it/s]
```



We have started with hyperparameter alpha with as low as 0.00001 to 1000. Since it is difficult to plot the given range we have used log alphas on x-axis and Auc on y axis as shown in the plot.

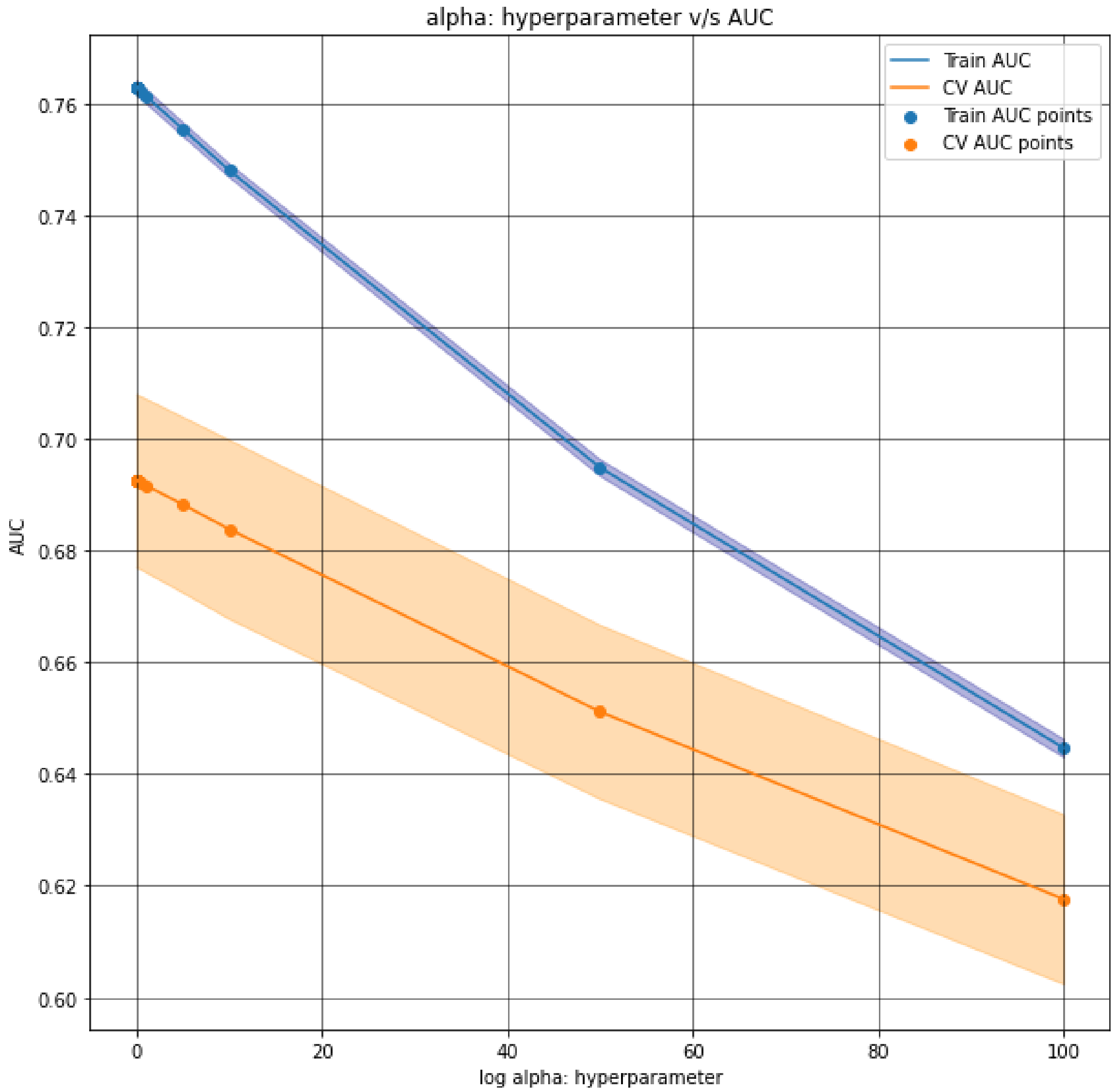
One of the main reason for using log scale is log scales allow a large range to be displayed without small values being compressed down into bottom of the graph.

we observe that as log alpha approaches close to 5, both train AUC and CV AUC lines converge

Using this plot we see after alpha=100 both lines converge at a much higher rate

In [27]:

```
# Graph between alpha and AUC
alphas = [0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1, 0.5, 1, 5, 10,
50, 100]
plt.figure(figsize=(10,10))

plt.plot(alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(alphas, train_auc - train_auc_std, train_auc + train_auc_
std, alpha=0.3, color='darkblue')

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

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


plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
print (train_auc)
print (cv_auc)
```




```
[0.76297605 0.76296781 0.76297226 0.76295267 0.76296492 0.76287239
 0.76294195 0.76279176 0.76217451 0.76141926 0.75546665 0.74809772
 0.69489149 0.6446869 ]
[0.69255442 0.6925539  0.69255427 0.69255581 0.69255419 0.69251892
 0.69254954 0.69247741 0.69212957 0.69169575 0.68824738 0.68384321
 0.65114865 0.6176183 ]
```

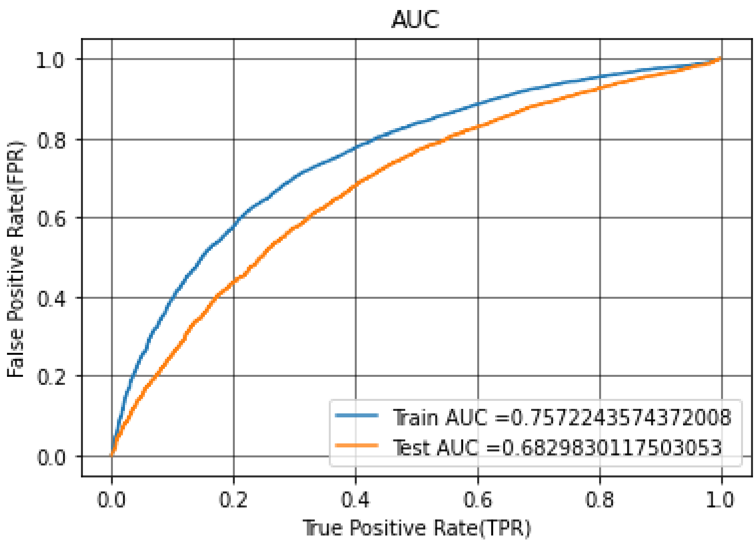
Train Model with best alpha

In [29]:

```
best_alpha=.0001
```

In [30]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.ht
ml#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

nb = MultinomialNB(alpha = best_alpha, class_prior=[0.5,0.5])

nb.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
es of the positive class
# not the predicted outputs

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("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



For BoW model and alpha=.0001 we are getting train AUC = .75 and test AUC=.68

Confusion Matrix

In [31]:

```
# writing our own function for predict, with defined thresold we will pick a thr
eshold that will give the least fpr*(1-tpr)
def predict(proba, threshold, fpr, tpr):

    t = threshold[np.argmax(fpr*(1-tpr))]      # (tpr*(1-fpr)) will be maximum i
f 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))
    predictions = []
    for i in proba:
        if i >= t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [32]:

```
# In confusion matrix we need y actual and y predicted (not probability score) s
o we are first converting probability score into 0/1 using predict method
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr,
train_tpr)))
```

```
====================================================================
================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4905655142900359 for threshold 0.
261
[[ 2268  1327]
 [ 4705 14145]]
```

In [33]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pr
ed, tr_thresholds, train_fpr, train_tpr)), range(2),range(2))
```

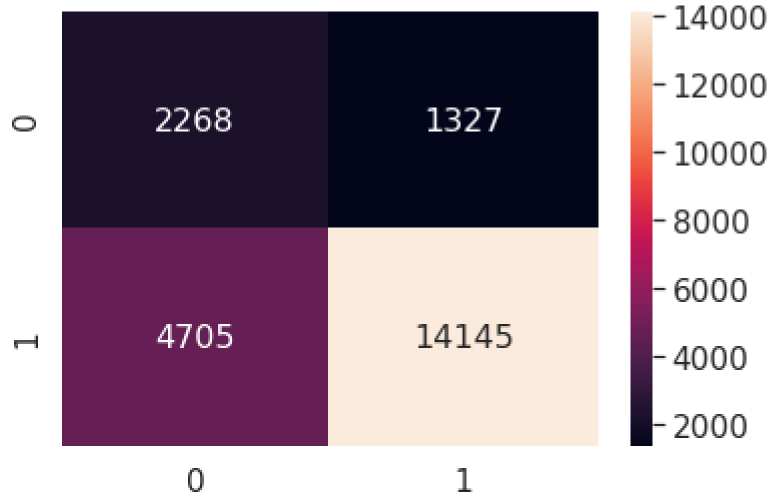
```
the maximum value of tpr*(1-fpr) 0.4905655142900359 for threshold 0.
261
```

In [34]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fefce80e7d0>
```



Summary on Train data: In the following confusion matrix we observe that the model has 14145-TP and 2268-TN, 4705-FP, 1327-FN

In [35]:

```
# test data
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, tes
t_tpr)))
```

```
====================================================================
================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.40903015543510474 for threshold
0.8
[[1760  882]
 [5386 8472]]
```

In [36]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred,
tr_thresholds, test_fpr, test_tpr)), range(2),range(2))
```

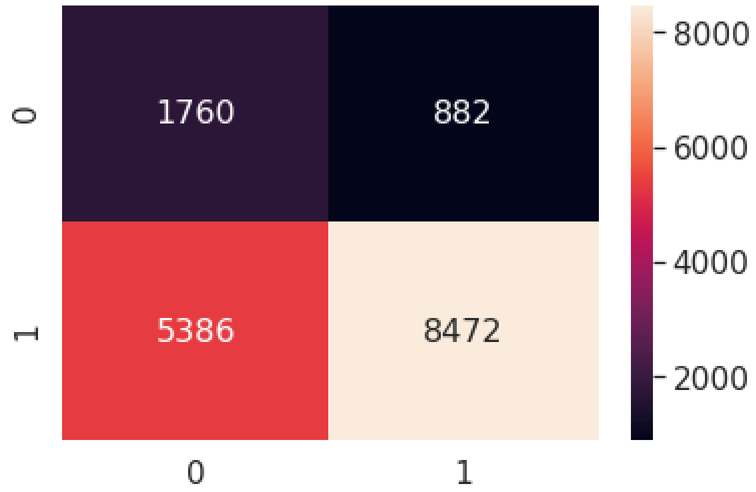
```
the maximum value of tpr*(1-fpr) 0.40903015543510474 for threshold
0.8
```

In [37]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[37]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fefdfa1ddd0>
```



Summary on test data: There are 8472-TP, 1760-TN, 882-FN, 5386-FP

In [38]:

```
nb_bow = MultinomialNB(alpha = 0.0001, class_prior = [0.5,0.5])
nb_bow.fit(X_tr, y_train)
```

Out[38]:

```
MultinomialNB(alpha=0.0001, class_prior=[0.5, 0.5], fit_prior=True)
```

In [39]:

```
# collecting feature names for BOW dataset1, adding to end of list by concatenin
g features

bow_features_names = []

for feat in vectorizer_essay.get_feature_names() :
    bow_features_names.append(feat)

for feat in vectorizer_clean.get_feature_names() :
    bow_features_names.append(feat)

for feat in vectorizer_cleansub.get_feature_names() :
    bow_features_names.append(feat)

for feat in vectorizer_grade.get_feature_names() :
    bow_features_names.append(feat)

for feat in vectorizer_prefix.get_feature_names() :
    bow_features_names.append(feat)

for feat in vectorizer_state.get_feature_names() :
    bow_features_names.append(feat)

#append numerical features name:
bow_features_names.append("price")
bow_features_names.append("teacher_number_of_previously_posted_projects")

print (len(bow_features_names))
```

```
5097
```

In [40]:

```
# print 20 Positive features for BOW dataset1. copied from https://stackoverflo
w.com/questions/16486252/is-it-possible-to-use-argsort-in-descending-order
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()[::-1][:5101]

for i in pos_class_prob_sorted[:20]:
    print(bow_features_names[i])
```

```
students
school
my
learning
classroom
the
not
they
learn
my students
help
price
many
nannan
work
we
reading
need
use
day
```

In [41]:

```
#print 20 Negative features for BOW set1 dataset. copied from https://stackoverf
low.com/questions/16486252/is-it-possible-to-use-argsort-in-descending-order
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()[::-1][:5101]
for i in neg_class_prob_sorted[0:20]:
    print(bow_features_names[i])
```

```
students
school
learning
my
classroom
not
learn
help
they
my students
the
price
many
nannan
need
we
work
come
reading
year
```

Summary: Words like use is present in positive class but not in negative class

Few words are similar but their relative ordering is different between the two sets

Set 2: categorical, numerical features + preprocessed_essay (TFIDF)

In [42]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

print("Before tfidf:")
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vectorizer_tfidf_essay = TfidfVectorizer(min_df=10,max_features=5000) #Consideri
ng top 5000 features
vectorizer_tfidf_essay.fit(X_train['essay'].values)

X_train_essay_tfidf = vectorizer_tfidf_essay.transform(X_train['essay'].values)
X_cv_essay_tfidf = vectorizer_tfidf_essay.transform(X_cv['essay'].values)
X_test_essay_tfidf = vectorizer_tfidf_essay.transform(X_test['essay'].values)

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

```
Before tfidf:
(22445, 8) (22445,)
(11055, 8) (11055,)
(16500, 8) (16500,)
====================================================================
================================
After tfidf
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
====================================================================
================================
```

In [44]:

```
# Combine all features
X_tr = hstack((X_train_essay_tfidf, X_train_state_ohe, X_train_teacher_ohe, X_tr
ain_grade_ohe, X_train_clean_ohe, X_train_cleansub_ohe, X_train_price_norm, X_tr
ain_teacherno_norm)).tocsr()
X_cr = hstack((X_cv_essay_tfidf, X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_oh
e, X_cv_clean_ohe, X_cv_cleansub_ohe, X_cv_price_norm, X_cv_teacherno_norm)).toc
sr()
X_te = hstack((X_test_essay_tfidf, X_test_state_ohe, X_test_teacher_ohe, X_test_
grade_ohe, X_test_clean_ohe, X_test_cleansub_ohe, X_test_price_norm, X_test_teac
herno_norm)).tocsr()

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

```
Final Data matrix
(22445, 5097) (22445,)
(11055, 5097) (11055,)
(16500, 5097) (16500,)
====================================================================
================================
```

Apply NB on dataset2: TFIDF

In [45]:

```
# function to return the probability score
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates 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 cr_loop will be 49041 - 49041%
1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [46]:

```
# The roc_auc_score function computes the area under the receiver operating char
acteristic (ROC) curve,
# By computing the area under the roc curve, the curve information is summarized
in one number.
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
log_alphas = []

alphas = [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(alphas):
    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)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates 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))

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)
```

```
100%|██████████| 14/14 [00:01<00:00,  8.59it/s]
100%|██████████| 14/14 [00:00<00:00, 5790.38it/s]
```

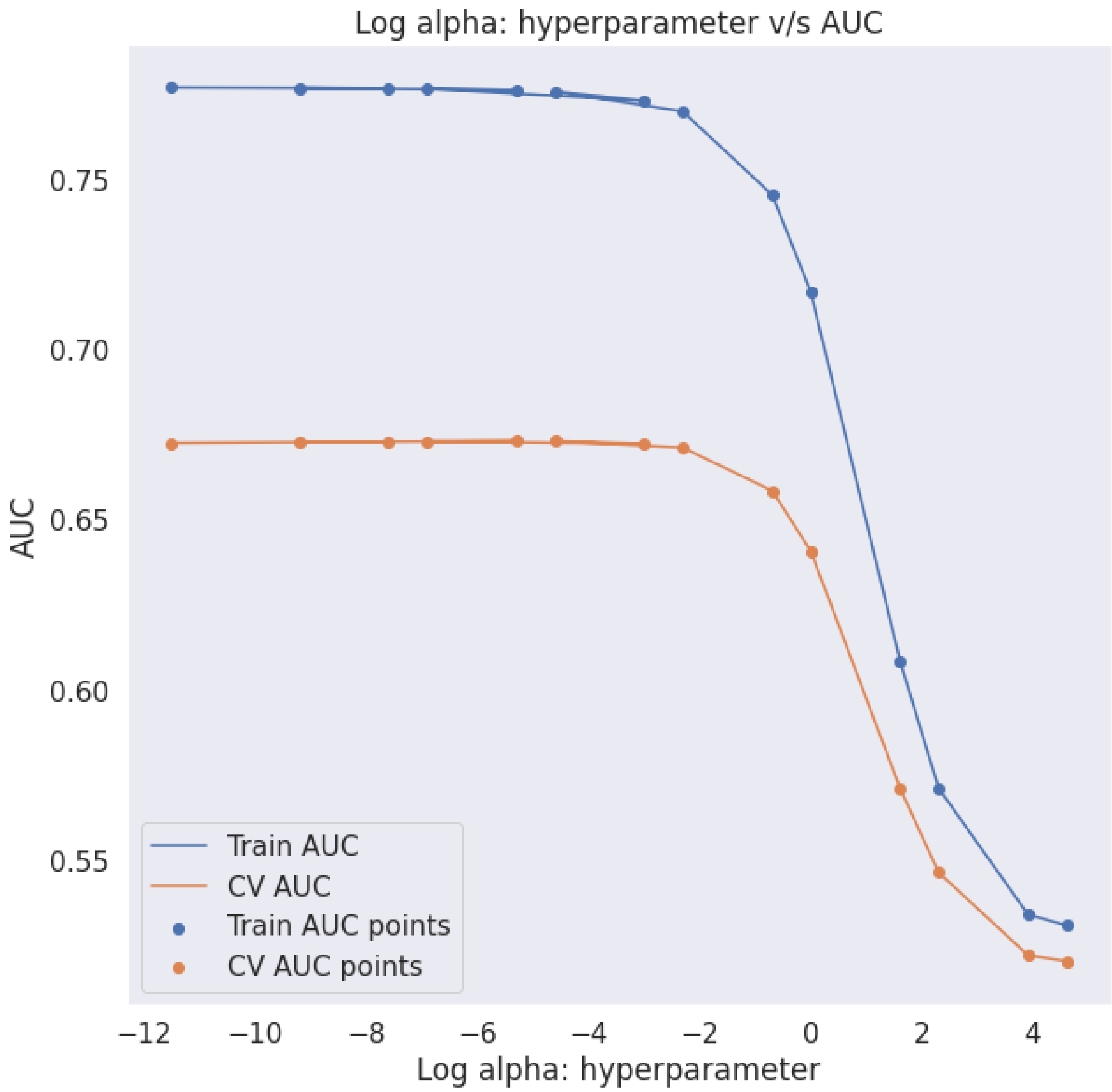
In [47]:

```
print (train_auc)
print (cv_auc)
```

```
[0.77725027761074, 0.777043934435906, 0.7771675366981108, 0.77650193
49745261, 0.7769504801466818, 0.7733405444490764, 0.776079273084117,
0.7701705064874217, 0.7456314583694565, 0.7170882341005596, 0.608444
5165883946, 0.5710121115755378, 0.5339026868292611, 0.53064728125933
82]
[0.6726040116949457, 0.6730192978773247, 0.6728295136131723, 0.67319
44786713275, 0.6730962703345715, 0.672392565616738, 0.67315949119076
09, 0.671250209164286, 0.6585591242785733, 0.6408163339813623, 0.570
988746200816, 0.5462391500780372, 0.5218644372035571, 0.520137090076
0294]
```

In [49]:

```
# plot the figure of log alpha (hyper-paramater) and AUC
plt.figure(figsize=(10, 10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("Log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("Log alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```

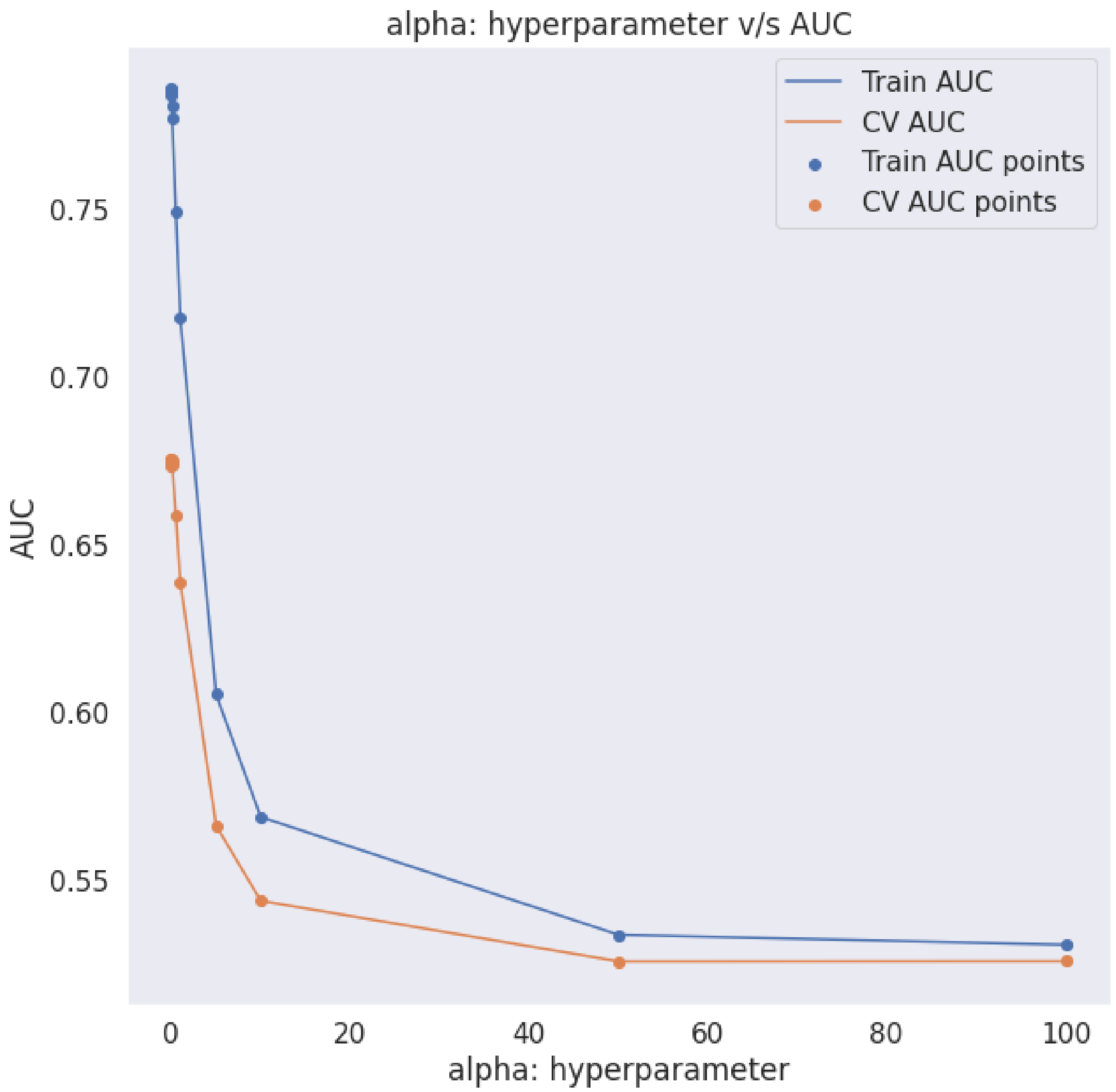


We have started with hyperparameter alpha with as low as 0.00001 to 1000. Since it is difficult to plot the given range we have used log alphas on x-axis and Auc on y axis as shown in the plot.

One of the main reason for using log scale is log scales allow a large range to be displayed without small values being compressed down into bottom of the graph.

In [51]:

```
# just to see how graph between alpha and AUC looks like
plt.figure(figsize=(10, 10))
plt.plot(alphas, train_auc, label='Train AUC')
plt.plot(alphas, cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid()
plt.show()
```



In [50]:

```
#k fold CV
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB(class_prior=[0.5,0.5])

parameters = {'alpha': [0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1,
0.5, 1, 5, 10, 50, 100]}

clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc',return_train_score=
True,verbose=2)

clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
Fitting 10 folds for each of 14 candidates, totalling 140 fits
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurr
ent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaini
ng:    0.0s
```

```
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=1e-05
....................................................
[CV] .................................... alpha=1e-05, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0005
...................................................
[CV] ................................... alpha=0.0005, total=   0.
```

```
0s
[CV] alpha=0.0005 .....................................................
[CV] ...................................... alpha=0.0005, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.0001 .....................................................
[CV] ...................................... alpha=0.0001, total=   0.
0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005 ......................................................
[CV] ....................................... alpha=0.005, total=   0.
0s
```

```
[CV] alpha=0.005
....................................................
[CV] ..................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005
....................................................
[CV] ..................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005
....................................................
[CV] ..................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005
....................................................
[CV] ..................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005
....................................................
[CV] ..................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.005
....................................................
[CV] ..................................... alpha=0.005, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
....................................................
[CV] ..................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.001
```

```
......................................................
[CV] ...................................... alpha=0.001, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.05
.......................................................
[CV] ....................................... alpha=0.05, total=   0.
0s
[CV] alpha=0.01
.......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
.......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
.......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
.......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
.......................................................
```

```
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.01
......................................................
[CV] ....................................... alpha=0.01, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
0s
[CV] alpha=0.1
.......................................................
[CV] ........................................ alpha=0.1, total=   0.
```

```
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=0.5
........................................................
[CV] ......................................... alpha=0.5, total=   0.
0s
[CV] alpha=1
..........................................................
[CV] ........................................... alpha=1, total=   0.
0s
[CV] alpha=1
..........................................................
[CV] ........................................... alpha=1, total=   0.
0s
[CV] alpha=1
..........................................................
[CV] ........................................... alpha=1, total=   0.
0s
[CV] alpha=1
..........................................................
[CV] ........................................... alpha=1, total=   0.
0s
[CV] alpha=1
..........................................................
[CV] ........................................... alpha=1, total=   0.
0s
```

```
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=1
.........................................................
[CV] .......................................... alpha=1, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=5
.........................................................
[CV] .......................................... alpha=5, total=   0.
0s
[CV] alpha=10
```

```
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=10
........................................................
[CV] ......................................... alpha=10, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
[CV] ......................................... alpha=50, total=   0.
0s
[CV] alpha=50
........................................................
```

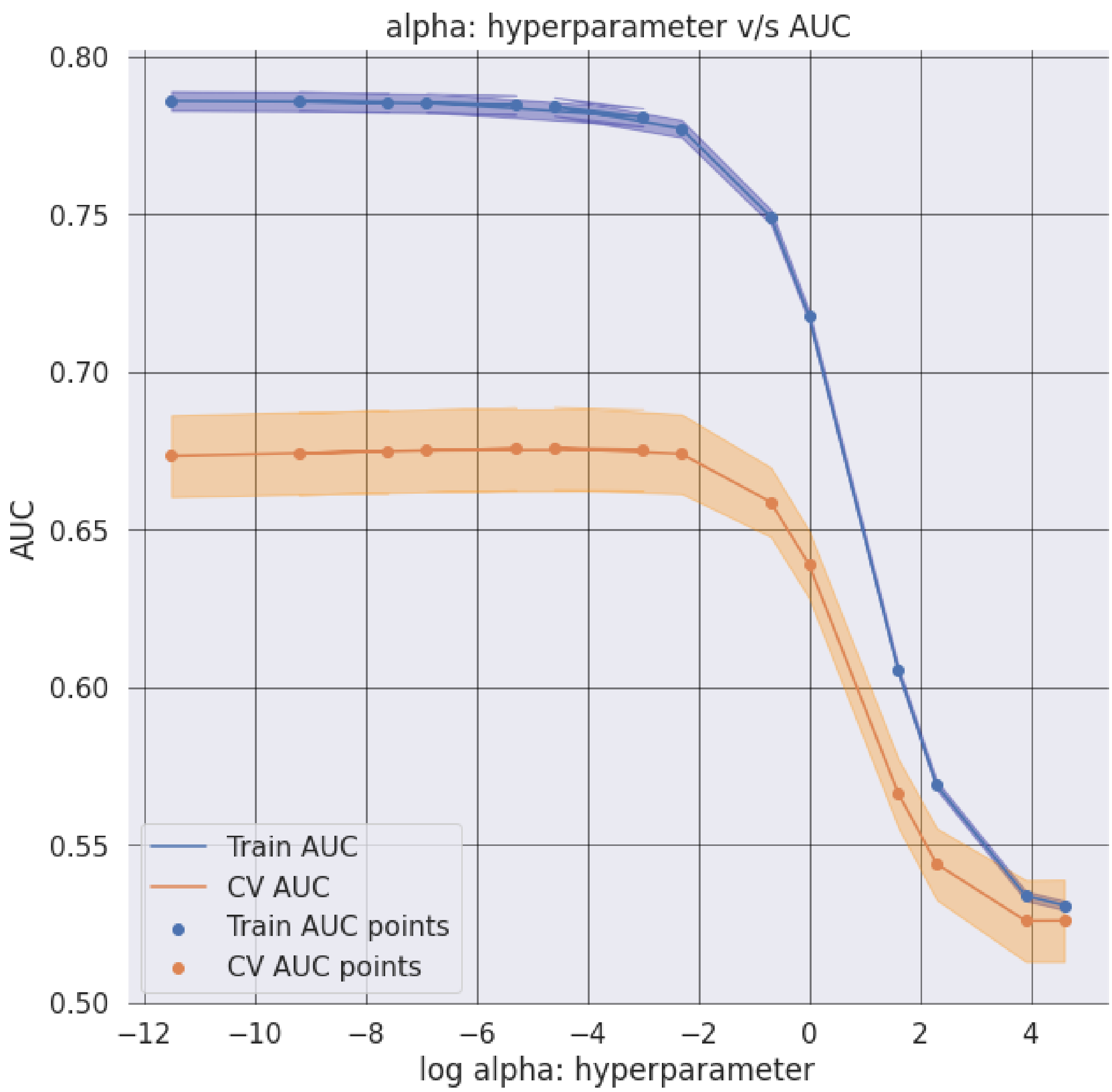
```
[CV] .......................................... alpha=50, total=   0.
0s
[CV] alpha=50
.........................................................
[CV] .......................................... alpha=50, total=   0.
0s
[CV] alpha=50
.........................................................
[CV] .......................................... alpha=50, total=   0.
0s
[CV] alpha=50
.........................................................
[CV] .......................................... alpha=50, total=   0.
0s
[CV] alpha=50
.........................................................
[CV] .......................................... alpha=50, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
[CV] alpha=100
........................................................
[CV] ......................................... alpha=100, total=   0.
0s
```

```
[Parallel(n_jobs=1)]: Done 140 out of 140 | elapsed:    8.6s finishe
d
```

In [52]:

```
# plot the graph between log alpha and AUC
alphas = [0.00001, 0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1, 0.5, 1, 5, 10,
50, 100]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(10,10))

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas, train_auc - train_auc_std, train_auc + train_
auc_std, alpha=0.3, color='darkblue')

plt.plot(log_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas, cv_auc - cv_auc_std, cv_auc + cv_auc_std, alp
ha=0.3, color='darkorange')

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


plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
100%|██████████| 14/14 [00:00<00:00, 2276.95it/s]
```

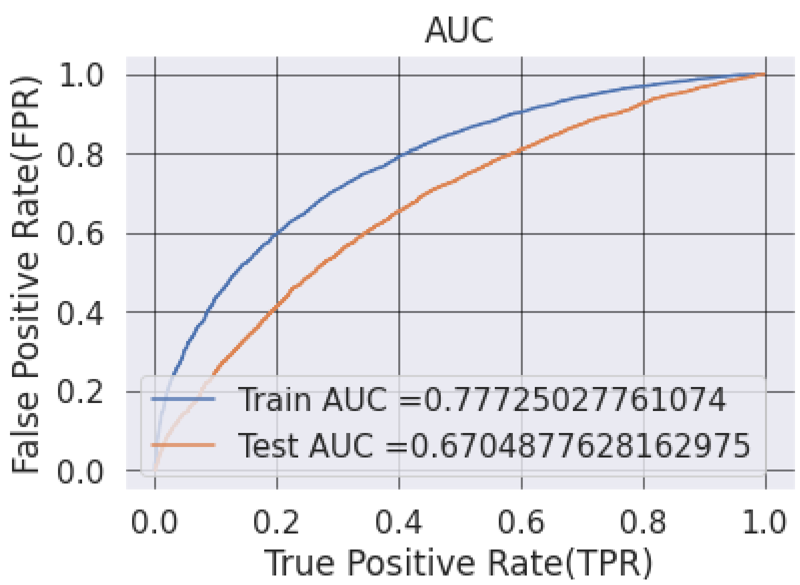


In [53]:

```
best_alpha=.00001
```

In [55]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.ht
ml#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

nb = MultinomialNB(alpha = best_alpha, class_prior=[0.5,0.5])

nb.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimat
es of the positive class
# not the predicted outputs

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("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



For TFIDF model and alpha=.0001 we are getting train AUC = .78 and test AUC=.67

Confusion matrix

In [56]:

```
# writing our own function for predict, with defined thresold we will pick a thr
eshold that will give the least fpr*(1-tpr)
def predict(proba, threshold, fpr, tpr):

    t = threshold[np.argmax(fpr*(1-tpr))]      # (tpr*(1-fpr)) will be maximum i
f 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))
    predictions = []
    for i in proba:
        if i >= t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [57]:

```
# In confusion matrix we need y actual and y predicted (not probability score) s
o we are first converting probability score into 0/1 using predict method
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr,
train_tpr)))
```

```
====================================================================
================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4993767500544154 for threshold 0.
469
[[ 2243  1352]
 [ 4386 14464]]
```

In [58]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pr
ed, tr_thresholds, train_fpr, train_tpr)), range(2),range(2))
```

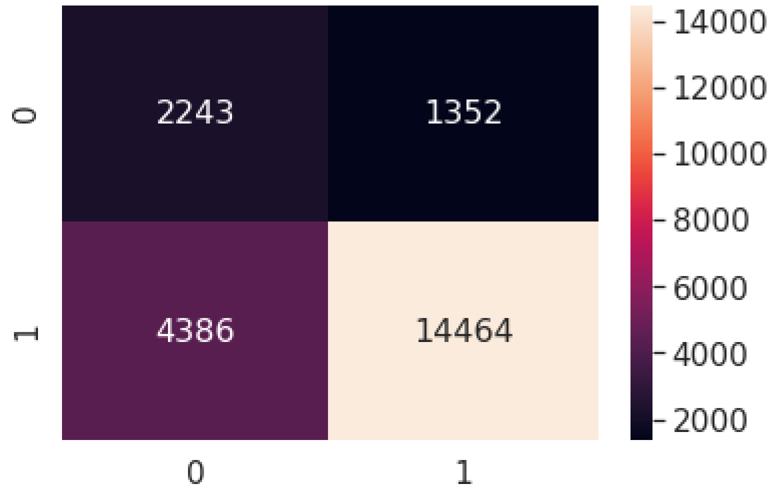
```
the maximum value of tpr*(1-fpr) 0.4993767500544154 for threshold 0.
469
```

In [59]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[59]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fefd89fbbd0>
```



Summary: TP: 14464, TN: 2243, FN: 1352, FP: 4386

In [60]:

```
# test data
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, tes
t_tpr)))
```

```
====================================================================
================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3940482512744984 for threshold 0.
507
[[1541 1101]
 [4587 9271]]
```

In [61]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred,
tr_thresholds, test_fpr, test_tpr)), range(2),range(2))
```

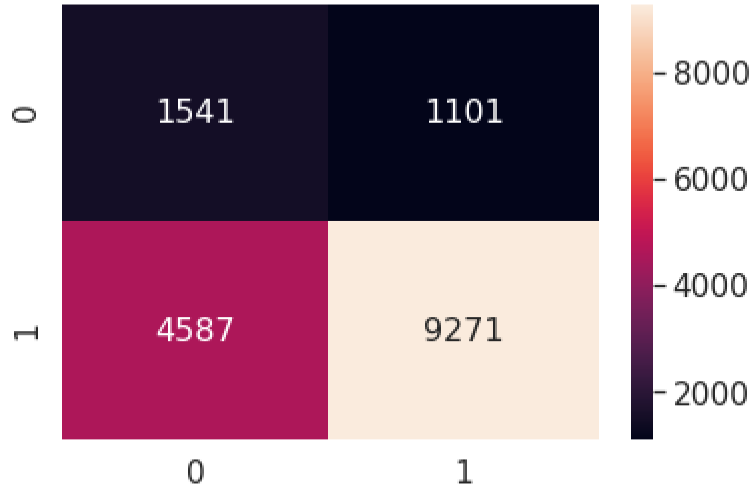
```
the maximum value of tpr*(1-fpr) 0.3940482512744984 for threshold 0.
507
```

In [62]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fefeba88750>
```



Summary: TP: 9271, TN: 1541, FN: 1101, FP: 4587

In [64]:

```
nb_tfidf = MultinomialNB(alpha = .00001, class_prior=[0.5,0.5])
nb_tfidf.fit(X_tr, y_train)
```

Out[64]:

```
MultinomialNB(alpha=1e-05, class_prior=[0.5, 0.5], fit_prior=True)
```

In [66]:

```
# collecting feature names for BOW dataset1, adding to end of list by concatenin
g features
tfidf_features_names = []

for feat in vectorizer_essay.get_feature_names() :
    tfidf_features_names.append(feat)

for feat in vectorizer_clean.get_feature_names() :
    tfidf_features_names.append(feat)

for feat in vectorizer_cleansub.get_feature_names() :
    tfidf_features_names.append(feat)

for feat in vectorizer_grade.get_feature_names() :
    tfidf_features_names.append(feat)

for feat in vectorizer_prefix.get_feature_names() :
    tfidf_features_names.append(feat)

for feat in vectorizer_state.get_feature_names() :
    tfidf_features_names.append(feat)

#append numerical features name:
tfidf_features_names.append("price")
tfidf_features_names.append("teacher_number_of_previously_posted_projects")

print (len(tfidf_features_names))
```

```
5097
```

In [67]:

```
#print 20 Positive features for BOW set1 dataset. copied from https://stackoverflow.com/questions/16486252/is-it-possible-to-use-argsort-in-descending-order
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()[::-1][:5101]

for i in pos_class_prob_sorted[:20]:
    print(bow_features_names[i])
```

```
students
school
my
learning
classroom
the
not
they
learn
my students
help
price
many
nannan
work
we
reading
need
use
day
```

In [68]:

```
#print 20 Negative features for BOW set1 dataset. copied from https://stackoverflow.com/questions/16486252/is-it-possible-to-use-argsort-in-descending-order
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()[::-1][:5101]
for i in neg_class_prob_sorted[0:20]:
    print(bow_features_names[i])
```

```
students
school
learning
my
classroom
not
learn
help
they
my students
the
price
many
nannan
need
we
work
come
reading
year
```

Summary: Words like use, day is present in positive class but not in negative class

Few words are similar but their relative ordering is different between the two sets

In [69]:

```
# compare models using Prettytable library http://zetcode.com/python/prettytabl
e/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha:Hyper Parameter", " Test AUC"]
x.add_row(["BOW", "Naive Bayes", 0.0001, 0.68])
x.add_row(["TFIDF", "Naive Bayes", 0.00001, 0.67])
print(x)
```

```
+------------+-------------+-----------------------+-----------+
| Vectorizer |    Model    | Alpha:Hyper Parameter |  Test AUC |
+------------+-------------+-----------------------+-----------+
|    BOW     | Naive Bayes |         0.0001        |    0.68   |
|   TFIDF    | Naive Bayes |         1e-05         |    0.67   |
+------------+-------------+-----------------------+-----------+
```