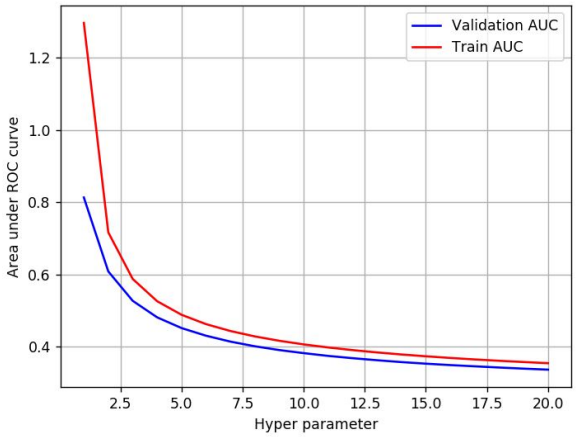
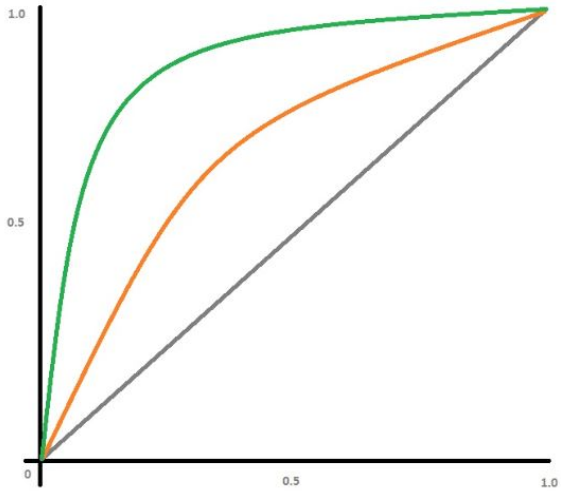
Assignment 6: Apply NB

1. **Apply Multinomial NB on these feature sets**
    - Set 1: categorical, numerical features + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)
2. **The hyper paramter tuning(find best alpha:smoothing parameter)**
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - find the best hyper paramter using k-fold cross validation(use GridsearchCV or RandomsearchCV)/simple cross validation data (write for loop to iterate over hyper parameter values)
    -
3. **Representation of results**
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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



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

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

4. fine the top 20 features from either from feature Set 1 or feature Set 2 using absolute values of `feature_log_prob_ ` parameter of `MultinomialNB` (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names
5. You need to summarize the results at the end of the notebook, summarize it in the table format

Vectorizer	Model	Hyper parameter	AUC
BOW	Brute	7	0.78
TFIDF	Brute	12	0.79

2. Naive Bayes

1.1 Loading Data

In [69]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from chart_studio import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

In [70]:

```
project_data = pd.read_csv('../train_data.csv')
resource_data = pd.read_csv('../resources.csv')
```

In [71]:

```
project_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109248 entries, 0 to 109247
Data columns (total 17 columns):
Unnamed: 0                                      109248 non-null int64
id                                              109248 non-null object
teacher_id                                      109248 non-null object
teacher_prefix                                  109245 non-null object
school_state                                    109248 non-null object
project_submitted_datetime                      109248 non-null object
project_grade_category                          109248 non-null object
project_subject_categories                      109248 non-null object
project_subject_subcategories                   109248 non-null object
project_title                                   109248 non-null object
project_essay_1                                 109248 non-null object
project_essay_2                                 109248 non-null object
project_essay_3                                 3758 non-null object
project_essay_4                                 3758 non-null object
project_resource_summary                        109248 non-null object
teacher_number_of_previously_posted_projects    109248 non-null int64
project_is_approved                             109248 non-null int64
dtypes: int64(3), object(14)
memory usage: 14.2+ MB
```

In [72]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
project_data.project_is_approved.value_counts()
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix'
'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

Out[72]:

```
1    92706
0    16542
Name: project_is_approved, dtype: int64
```

Pre-processing of project_subject_categories

In [73]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

Pre-processing of project_subject_subcategories

In [74]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())


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

Pre-processing of project grade categories

In [75]:

```
project_data.groupby(['project_grade_category'])['project_grade_category'].count()
```

Out[75]:

```
project_grade_category
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Grades PreK-2    44225
Name: project_grade_category, dtype: int64
```

In [76]:

```
project_data['project_grade_category'][project_data['project_grade_category'].isnull()==True]
```

Out[76]:

```
Series([], Name: project_grade_category, dtype: object)
```

In [77]:

```
project_grade_category = list(project_data['project_grade_category'].values)

project_grade_category_list = []
for i in project_grade_category:
    temp = ""
    temp = i.split(' ')
    temp = i.replace('Grades ','')
    project_grade_category_list.append(temp)

project_data['clean_project_grade_category'] = project_grade_category_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_project_grade_category'].values:
    my_counter.update(word.split())

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

In [78]:

```
sorted_project_grade_category_dict.keys()
```

Out[78]:

```
dict_keys(['9-12', '6-8', '3-5', 'PreK-2'])
```

In [79]:

```
project_data.groupby(['clean_project_grade_category'])['clean_project_grade_category'].count()
```

Out[79]:

```
clean_project_grade_category
3-5       37137
6-8       16923
9-12      10963
PreK-2    44225
Name: clean_project_grade_category, dtype: int64
```

Pre-processing of teacher prefix

In [80]:

```
project_data.groupby(['teacher_prefix'])['teacher_prefix'].count()
```

Out[80]:

```
teacher_prefix
Dr.           13
Mr.        10648
Mrs.       57269
Ms.        38955
Teacher     2360
Name: teacher_prefix, dtype: int64
```

In [81]:

```
project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

Out[81]:

```
7820     NaN
30368    NaN
57654    NaN
Name: teacher_prefix, dtype: object
```

In [82]:

```
project_data['teacher_prefix'].fillna(project_data['teacher_prefix'].mode()[0],inplace=
True)
```

In [83]:

```
project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

Out[83]:

```
Series([], Name: teacher_prefix, dtype: object)
```

In [84]:

```
project_data['teacher_prefix'].unique()
```

Out[84]:

```
array(['Mrs.', 'Mr.', 'Ms.', 'Teacher', 'Dr.'], dtype=object)
```

In [85]:

```
teacher_prefix = list(project_data['teacher_prefix'].values)

teacher_prefix_list = []
for i in teacher_prefix:
    temp = ""
    temp = i.split('.')
    temp = i.replace('.','')
    teacher_prefix_list.append(temp)

project_data['clean_teacher_prefix'] = teacher_prefix_list
project_data.drop(['teacher_prefix'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_teacher_prefix'].values:
    my_counter.update(word.split())

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

In [86]:

```
sorted_teacher_prefix_dict.keys()
```

Out[86]:

```
dict_keys(['Dr', 'Teacher', 'Mr', 'Ms', 'Mrs'])
```

In [87]:

```
project_data.groupby(['clean_teacher_prefix'])['clean_teacher_prefix'].count()
```

Out[87]:

```
clean_teacher_prefix
Dr            13
Mr         10648
Mrs        57272
Ms         38955
Teacher     2360
Name: clean_teacher_prefix, dtype: int64
```

preprocessing of school state

In [88]:

```
project_data['school_state'].unique()
```

Out[88]:

```
array(['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY',
       'OK', 'MA', 'NV', 'OH', 'PA', 'AL', 'LA', 'VA', 'AR', 'WA', 'WV',
       'ID', 'TN', 'MS', 'CO', 'UT', 'IL', 'MI', 'HI', 'IA', 'RI', 'NJ',
       'MO', 'DE', 'MN', 'ME', 'WY', 'ND', 'OR', 'AK', 'MD', 'WI', 'SD',
       'NE', 'NM', 'DC', 'KS', 'MT', 'NH', 'VT'], dtype=object)
```

In [89]:

```
project_data['school_state'][project_data['school_state'].isnull()==True]
```

Out[89]:

```
Series([], Name: school_state, dtype: object)
```

In [90]:

```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())

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

In [91]:

```
sorted_school_state_dict.keys()
```

Out[91]:

```
dict_keys(['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'W
V', 'ME', 'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'OR', 'K
Y', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'O
K', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'N
C', 'FL', 'NY', 'TX', 'CA'])
```

Text preprocessing

Pre-processing of essay

In [92]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [93]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [94]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [95]:

```
# Combining all the above stundents
def Text_cleaner(data):
    from tqdm import tqdm
    preprocessed_essays = []
    # tqdm is for printing the status bar
    for sentance in tqdm(data.values):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\"', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('nan',' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
        preprocessed_essays.append(sent.lower().strip())
    return preprocessed_essays
```

In [96]:

```
# after preprocesing
preprocessed_essays=Text_cleaner(project_data['essay'])
```

```
100%|████████████████████████████████████████████████████████████████████████
████| 109248/109248 [02:21<00:00, 771.84it/s]
```

In [97]:

```
project_data['preprocessed_essays'] = preprocessed_essays
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
project_data.drop(['essay'], axis=1, inplace=True)
```

Preprocessing of title

In [98]:

```
#convert all the words to lower case first and then remove the stopwords
for i in range(len(project_data['project_title'].values)):
    project_data['project_title'].values[i] = project_data['project_title'].values[i].l
ower()
```

In [99]:

```
# similarly you can preprocess the titles also

preprocessed_titles = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = sent.replace('nan',' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
██| 109248/109248 [00:06<00:00, 16963.93it/s]
```

In [100]:

```
preprocessed_titles[20000]
```

Out[100]:

```
'need move input'
```

In [101]:

```
#creating a new column with the preprocessed titles,useful for analysis
project_data['preprocessed_titles'] = preprocessed_titles
```

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

Firstly, we split the data into train and test in the 2:1 ratio

Secondly, we stratify/group the data using the column "project_is_approved"

In [102]:

```
project_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109248 entries, 0 to 109247
Data columns (total 15 columns):
Unnamed: 0                                        109248 non-null int64
id                                                109248 non-null object
teacher_id                                        109248 non-null object
school_state                                      109248 non-null object
project_submitted_datetime                        109248 non-null object
project_title                                     109248 non-null object
project_resource_summary                          109248 non-null object
teacher_number_of_previously_posted_projects      109248 non-null int64
project_is_approved                               109248 non-null int64
clean_categories                                  109248 non-null object
clean_subcategories                               109248 non-null object
clean_project_grade_category                      109248 non-null object
clean_teacher_prefix                              109248 non-null object
preprocessed_essays                               109248 non-null object
preprocessed_titles                               109248 non-null object
dtypes: int64(3), object(12)
memory usage: 12.5+ MB
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data (optional)
- quantity : numerical (optional)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

In [103]:

```
project_data = project_data.drop(['Unnamed: 0','project_submitted_datetime','project_re
source_summary','teacher_id'], axis = 1)
```

In [104]:

```
project_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 109248 entries, 0 to 109247
Data columns (total 11 columns):
id                                              109248 non-null object
school_state                                    109248 non-null object
project_title                                   109248 non-null object
teacher_number_of_previously_posted_projects    109248 non-null int64
project_is_approved                             109248 non-null int64
clean_categories                                109248 non-null object
clean_subcategories                             109248 non-null object
clean_project_grade_category                    109248 non-null object
clean_teacher_prefix                            109248 non-null object
preprocessed_essays                             109248 non-null object
preprocessed_titles                             109248 non-null object
dtypes: int64(2), object(9)
memory usage: 9.2+ MB
```

In [105]:

```
# train test split

from sklearn.model_selection import train_test_split

project_data_train, project_data_test, y_train, y_test = train_test_split(project_data,
project_data['project_is_approved'], test_size=0.33, stratify = project_data['project_i
s_approved'])
```

Now, we are checking the ratio of train, test division along with their size

In [106]:

```
print("Split ratio")
print('-'*50)
print('Train dataset:',len(project_data_train)/len(project_data)*100,'%\n','size:',len(
project_data_train))

print('Test dataset:',len(project_data_test)/len(project_data)*100,'%\n','size:',len(pr
oject_data_test))
```

```
Split ratio
--------------------------------------------------
Train dataset: 66.99985354422964 %
 size: 73196
Test dataset: 33.000146455770356 %
 size: 36052
```

In [107]:

```
#Features
project_data_train.drop(['project_is_approved'], axis=1, inplace=True)

project_data_test.drop(['project_is_approved'], axis=1, inplace=True)
```

In [ ]:

```
```

Preparing data for models

1.4 Make Data Model Ready: encoding numerical, categorical features

Vectorizing Categorical data

One hot encoding features

In [109]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(lowercase=False, binary=True)
vectorizer_cat.fit(project_data_train['clean_categories'].values) #fitting has to be on
Train data

train_categories_one_hot = vectorizer_cat.transform(project_data_train['clean_categorie
s'].values)

test_categories_one_hot = vectorizer_cat.transform(project_data_test['clean_categories'
].values)


print(vectorizer_cat.get_feature_names())
print("Shape of training data matrix after one hot encoding ",train_categories_one_hot.
shape)

print("Shape of test data matrix after one hot encoding ",test_categories_one_hot.shape
)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Lit
eracy_Language', 'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of training data matrix after one hot encoding  (73196, 9)
Shape of test data matrix after one hot encoding  (36052, 9)
```

In [110]:

```
# we use count vectorizer to convert the values into one
vectorizer_subcat = CountVectorizer(lowercase=False, binary=True)
vectorizer_subcat.fit(project_data_train['clean_subcategories'].values)


train_subcategories_one_hot = vectorizer_subcat.transform(project_data_train['clean_sub
categories'].values)

test_subcategories_one_hot = vectorizer_subcat.transform(project_data_test['clean_subca
tegories'].values)


print(vectorizer_subcat.get_feature_names())


print("Shape of train data matrix after one hot encoding ",train_subcategories_one_hot.
shape)

print("Shape of test data matrix after one hot encoding ",test_subcategories_one_hot.sh
ape)
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Governmen
t', 'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment',
'Economics', 'EnvironmentalScience', 'Extracurricular', 'FinancialLiterac
y', 'ForeignLanguages', 'Gym_Fitness', 'Health_LifeScience', 'Health_Welln
ess', 'History_Geography', 'Literacy', 'Literature_Writing', 'Mathematic
s', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'Performi
ngArts', 'SocialSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Wa
rmth']
Shape of train data matrix after one hot encoding  (73196, 30)
Shape of test data matrix after one hot encoding  (36052, 30)
```

In [111]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer_school_state = CountVectorizer()
vectorizer_school_state.fit(project_data_train['school_state'].values)



print(vectorizer_school_state.get_feature_names())

train_school_state_category_one_hot = vectorizer_school_state.transform(project_data_tr
ain['school_state'].values)


test_school_state_category_one_hot = vectorizer_school_state.transform(project_data_tes
t['school_state'].values)


print("Shape of train data matrix after one hot encoding ",train_school_state_category_
one_hot.shape)

print("Shape of test data matrix after one hot encoding ",test_school_state_category_on
e_hot.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi',
'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'm
o', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'o
k', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'w
i', 'wv', 'wy']
Shape of train data matrix after one hot encoding  (73196, 51)
Shape of test data matrix after one hot encoding  (36052, 51)
```

In [117]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys
()), lowercase=False, binary=True)
vectorizer.fit(project_data_train['clean_project_grade_category'].values) # fit has to
 happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
train_project_grade_category_one_hot = vectorizer.transform(project_data_train['clean_p
roject_grade_category'].values)
test_project_grade_category_one_hot = vectorizer.transform(project_data_test['clean_pro
ject_grade_category'].values)

print("After vectorizations")
print(train_project_grade_category_one_hot.shape, y_train.shape)
print(test_project_grade_category_one_hot.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(73196, 4) (73196,)
(36052, 4) (36052,)
['9-12', '6-8', '3-5', 'PreK-2']
====================================================================================
==========================
```

In [118]:

```
project_data['clean_project_grade_category'].unique()
```

Out[118]:

```
array(['PreK-2', '6-8', '3-5', '9-12'], dtype=object)
```

In [120]:

```
from collections import Counter
my_counter = Counter()
for word in project_data['clean_teacher_prefix'].values:
    if not isinstance(word, float):
      word = word.replace('.',' ')
      my_counter.update(word.split())

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

In [121]:

```
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerro
r-np-nan-is-an-invalid-document
#ValueError: np.nan is an invalid document, expected byte or unicode string.
vectorizer_prefix = CountVectorizer()
vectorizer_prefix.fit(project_data_train['clean_teacher_prefix'].values.astype("U"))


print(vectorizer_prefix.get_feature_names())


train_teacher_prefix_categories_one_hot = vectorizer_prefix.transform(project_data_trai
n['clean_teacher_prefix'].values.astype("U"))

test_teacher_prefix_categories_one_hot = vectorizer_prefix.transform(project_data_test[
'clean_teacher_prefix'].values.astype("U"))

print("Shape of train data matrix after one hot encoding ",train_teacher_prefix_categor
ies_one_hot.shape)

print("Shape of test data matrix after one hot encoding ",test_teacher_prefix_categorie
s_one_hot.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of train data matrix after one hot encoding  (73196, 5)
Shape of test data matrix after one hot encoding  (36052, 5)
```

1.3 Make Data Model Ready: encoding essay, project title

Vectorizing Text data

A) Bag of Words

In [123]:

```
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(project_data_train['preprocessed_essays'].values)  #Fitting ha
s to be on Train data


train_essay_bow = vectorizer_bow_essay.transform(project_data_train['preprocessed_essay
s'].values)

test_essay_bow = vectorizer_bow_essay.transform(project_data_test['preprocessed_essays'
].values)


print("Shape of train data matrix after one hot encoding ",train_essay_bow.shape)

print("Shape of test data matrix after one hot encoding ",test_essay_bow.shape)
```

```
Shape of train data matrix after one hot encoding  (73196, 14111)
Shape of test data matrix after one hot encoding  (36052, 14111)
```

In [124]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit_transform(project_data_train['preprocessed_titles'].values)
#Fitting has to be on Train data


train_title_bow = vectorizer_bow_title.transform(project_data_train['preprocessed_title
s'].values)

test_title_bow = vectorizer_bow_title.transform(project_data_test['preprocessed_titles'
].values)



print("Shape of train data matrix after one hot encoding ",train_title_bow.shape)

print("Shape of test data matrix after one hot encoding ",test_title_bow.shape)
```

```
Shape of train data matrix after one hot encoding  (73196, 2535)
Shape of test data matrix after one hot encoding  (36052, 2535)
```

B) TFIDF

In [125]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(project_data_train['preprocessed_essays'])      #Fitting has
to be on Train data

train_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_train['preprocessed_e
ssays'].values)

test_essay_tfidf = vectorizer_tfidf_essay.transform(project_data_test['preprocessed_ess
ays'].values)

print("Shape of train data matrix after one hot encoding ",train_essay_tfidf.shape)

print("Shape of test data matrix after one hot encoding ",test_essay_tfidf.shape)
```

```
Shape of train data matrix after one hot encoding  (73196, 14111)
Shape of test data matrix after one hot encoding  (36052, 14111)
```

In [126]:

```
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(project_data_train['preprocessed_titles'])      #Fitting has
to be on Train data

train_title_tfidf = vectorizer_tfidf_title.transform(project_data_train['preprocessed_t
itles'].values)

test_title_tfidf = vectorizer_tfidf_title.transform(project_data_test['preprocessed_tit
les'].values)

print("Shape of train data matrix after one hot encoding ",train_title_tfidf.shape)

print("Shape of test data matrix after one hot encoding ",test_title_tfidf.shape)
```

```
Shape of train data matrix after one hot encoding  (73196, 2535)
Shape of test data matrix after one hot encoding  (36052, 2535)
```

In [ ]:

```

```

Using Pretrained Models: Avg W2V

In [127]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
import pickle
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

Train essays

In [128]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(project_data_train['preprocessed_essays']): # for each review/sent
ence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_avg_w2v_essays.append(vector)

print(len(train_avg_w2v_essays))
print(len(train_avg_w2v_essays[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██████| 73196/73196 [00:44<00:00, 1646.37it/s]

73196
300
```

Test essays

In [129]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essays']): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_avg_w2v_essays.append(vector)

print(len(test_avg_w2v_essays))
print(len(test_avg_w2v_essays[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██████| 36052/36052 [00:22<00:00, 1627.69it/s]

36052
300
```

Train titles

In [130]:

```
# compute average word2vec for each review.
train_avg_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(project_data_train['preprocessed_titles']): # for each review/sent
ence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_avg_w2v_titles.append(vector)

print(len(train_avg_w2v_titles))
print(len(train_avg_w2v_titles[0]))
```

```
100%|████████████████████████████████████████████████████████████████
████| 73196/73196 [00:02<00:00, 29545.76it/s]

73196
300
```

Test titles

In [131]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_titles']): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_avg_w2v_titles.append(vector)

print(len(test_avg_w2v_titles))
print(len(test_avg_w2v_titles[0]))
```

```
100%|████████████████████████████████████████████████████████████████
████| 36052/36052 [00:01<00:00, 26099.90it/s]

36052
300
```

Using pretrained models : TFIDF Weighted W2V

Train essay

In [132]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_essays'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [133]:

```
# compute average word2vec for each review.
train_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(project_data_train['preprocessed_essays']): # for each review/sent
ence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_essays.append(vector)

print(len(train_tfidf_w2v_essays))
print(len(train_tfidf_w2v_essays[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██████| 73196/73196 [05:28<00:00, 222.80it/s]

73196
300
```

Test essay

In [134]:

```
# compute average word2vec for each review.
test_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(project_data_test['preprocessed_essays']): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_essays.append(vector)

print(len(test_tfidf_w2v_essays))
print(len(test_tfidf_w2v_essays[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██████| 36052/36052 [02:38<00:00, 227.45it/s]

36052
300
```

Train titles

In [135]:

```
# Similarly you can vectorize for title also
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_titles'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [136]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(project_data_train['preprocessed_titles']): # for each review/sent
ence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_titles.append(vector)

print(len(train_tfidf_w2v_titles))
print(len(train_tfidf_w2v_titles[0]))
```

```
100%|████████████████████████████████████████████████████████████████
████| 73196/73196 [00:05<00:00, 14417.05it/s]

73196
300
```

Test titles

In [137]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(project_data_test['preprocessed_titles']): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_titles.append(vector)

print(len(test_tfidf_w2v_titles))
print(len(test_tfidf_w2v_titles[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████
█████| 36052/36052 [00:03<00:00, 9836.32it/s]

36052
300
```

Vectorizing Numerical Features

Numerical Features present in dataset are :

1. Price
2. Number of Projects previously proposed by Teacher

1) Price

In [138]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_i
ndex()
```

In [139]:

```
project_data_train = pd.merge(project_data_train, price_data, on='id', how='left')

project_data_test = pd.merge(project_data_test, price_data, on='id', how='left')
```

In [ ]:

```
```

In [140]:

```
from sklearn.preprocessing import Normalizer
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer = Normalizer()
normalizer.fit(project_data_train['price'].values.reshape(1,-1))


price_normalized_train = normalizer.transform(project_data_train['price'].values.reshap
e(1, -1))

price_normalized_test = normalizer.transform(project_data_test['price'].values.reshape(
1, -1))
#reshaping again after normalization

price_normalized_train = price_normalized_train.reshape(-1, 1)
price_normalized_test =  price_normalized_test.reshape(-1, 1)

print('After normalization')
print(price_normalized_train.shape)

print(price_normalized_test.shape)
```

```
After normalization
(73196, 1)
(36052, 1)
```

2) No.of projects proposed by Teacher

In [141]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['teacher_number_of_previously_posted_projects'].value
s.reshape(1,-1))


previously_posted_projects_normalized_train = normalizer.transform(project_data_train[
'teacher_number_of_previously_posted_projects'].values.reshape(1, -1))

previously_posted_projects_normalized_test = normalizer.transform(project_data_test['te
acher_number_of_previously_posted_projects'].values.reshape(1, -1))

#reshaping again after normalization

previously_posted_projects_normalized_train = previously_posted_projects_normalized_tra
in.reshape(-1,1)
previously_posted_projects_normalized_test = previously_posted_projects_normalized_test
.reshape(-1,1)

print('After normalization')
print(previously_posted_projects_normalized_train.shape)

print(previously_posted_projects_normalized_test.shape)
```

```
After normalization
(73196, 1)
(36052, 1)
```

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

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

Applying Naive Bayes on BOW,

Set 1 : categorical, numerical features + preprocessed_essay (BOW)

In [142]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_bo
w, train_title_bow, train_school_state_category_one_hot,train_teacher_prefix_categories
_one_hot, previously_posted_projects_normalized_train, train_project_grade_category_one
_hot, price_normalized_train)).tocsr()

X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_bow, t
est_title_bow, test_school_state_category_one_hot, test_teacher_prefix_categories_one_h
ot, previously_posted_projects_normalized_test, test_project_grade_category_one_hot, pr
ice_normalized_test)).tocsr()

print(X_train.shape)

print(X_test.shape)
```

```
(73196, 16747)
(36052, 16747)
```

In [143]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your 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
```

2) a) Random Alpha Values (hyperparameter)

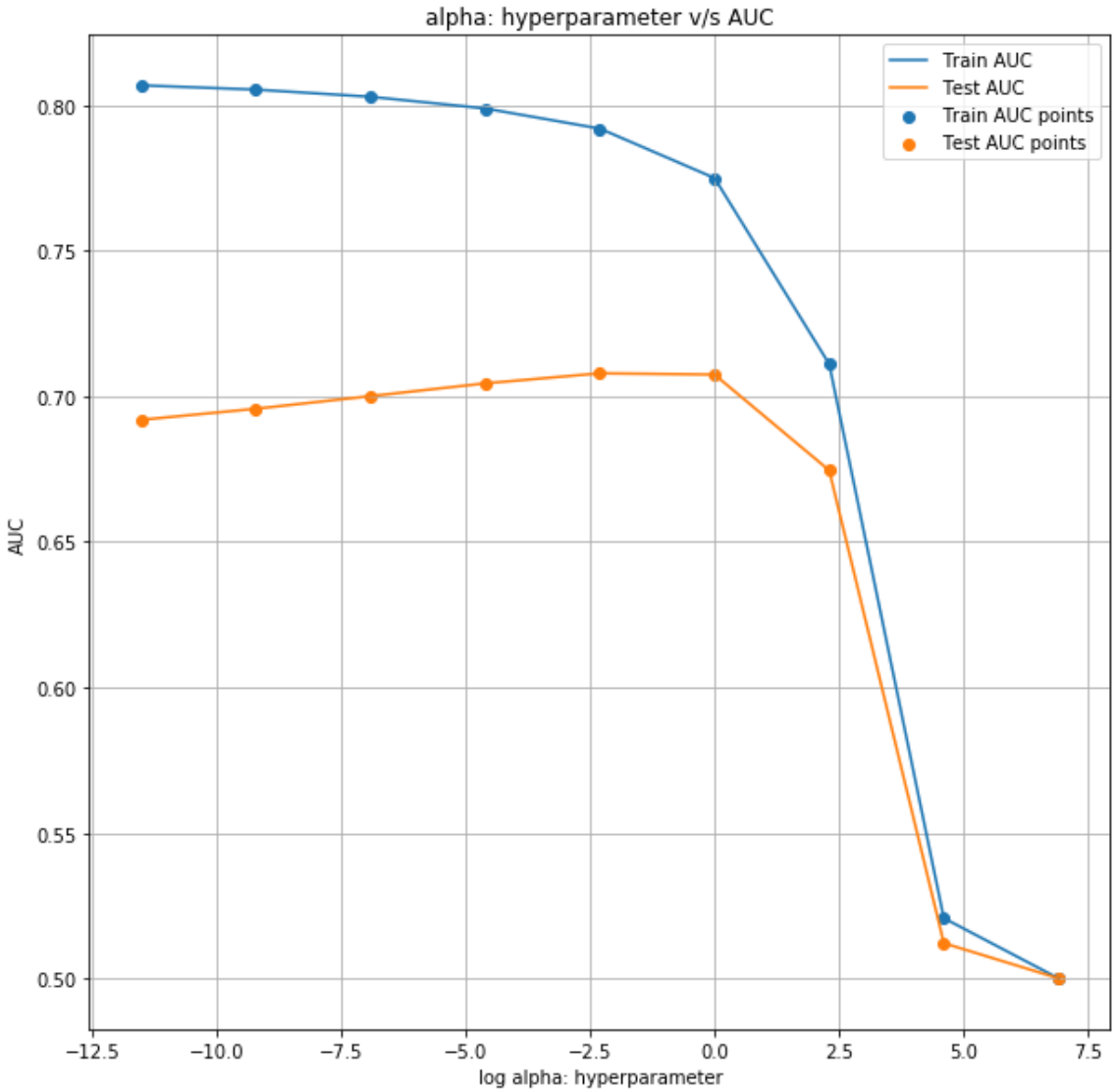
In [144]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
test_auc = []
log_alphas = []

alphas = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10,  100,  1000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior=[0.5,0.5])
    nb.fit(X_train, y_train)

    y_train_pred = batch_predict(nb, X_train)
    y_test_pred = batch_predict(nb, X_test)


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    test_auc.append(roc_auc_score(y_test, y_test_pred))

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

```
100%|██████████| 9/9 [00:04<00:00,  1.93it/s]
100%|██████████| 9/9 [00:00<00:00, 9056.80it/s]
```

In [145]:

```
plt.figure(figsize=(10,10))
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, test_auc, label='Test AUC')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, test_auc, label='Test AUC points')

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



Observation

1. We have started with hyperparameter alpha with as low as 0.0001 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.
2. 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.
3. we observe that as log alpha approaches close to 7 ,both train AUC and Test AUC lines converge
4. Using this plot we see after alpha=10 both lines converge at amuch higher rate

In [ ]:

```
```

b) GridSearch-cv using cv=10 (K fold cross Validation)

In [146]:

```
from sklearn.model_selection import GridSearchCV

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

parameters = {'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]}

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

clf.fit(X_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
test_auc = clf.cv_results_['mean_test_score']
test_auc_std= clf.cv_results_['std_test_score']
```

```
Fitting 10 folds for each of 11 candidates, totalling 110 fits
[CV] alpha=1e-05 .....................................................
[CV] ..................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.4s remaining:
0.0s
```

```
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
```

```
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
```

```
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
```

```
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.2s

[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:   44.8s finished
```

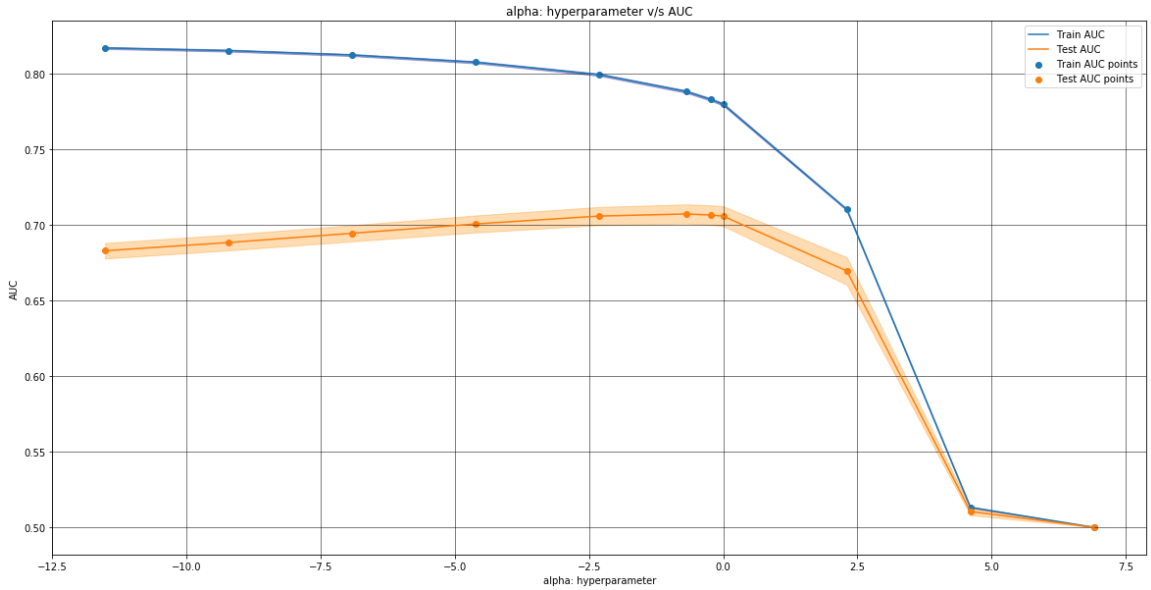
In [ ]:

```
```

In [147]:

```
alphas = [0.00001, 0.0001, 0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]
log_alphas =[]

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

plt.figure(figsize=(20,10))

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,a
lpha=0.3,color='darkblue')

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

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, test_auc, label='Test AUC points')


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

```
100%|██████████████████████████████████████████████████████████████████████████████
██████████████████| 11/11 [00:00<?, ?it/s]
```



In [ ]:

```
```

In [148]:

```
def pred_prob(clf, data):
    y_pred = []
    y_pred = clf.predict_proba(data)[:,1]
    return y_pred
```

c) Train model using the best hyperparameter value

1. Using bestparams attribute of gridsearch cv we can obtain the optimal value of alpha among the values we have selected
2. It simplifes our task and we can be rest assured that selected hyperparameter is most optimal one

In [149]:

```
#https://datascience.stackexchange.com/questions/21877/how-to-use-the-output-of-gridsea
rch
#choosing the best hyperparameter
clf.best_params_
```

Out[149]:

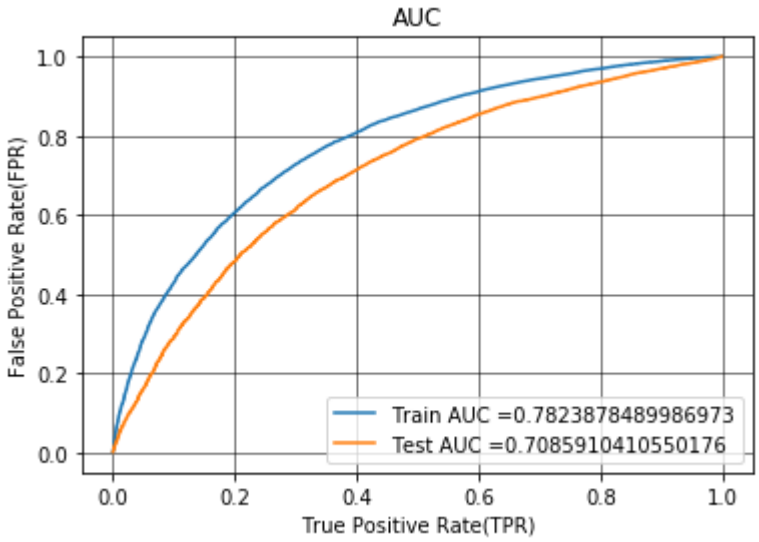
```
{'alpha': 0.5}
```

In [150]:

```
best_alpha1=clf.best_params_['alpha']
```

In [151]:

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

nb_bow = MultinomialNB(alpha = 0.5,class_prior=[0.5,0.5])

nb_bow.fit(X_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(nb_bow, X_train)
y_test_pred = batch_predict(nb_bow, X_test)

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()
```



Summary

For Bow model for alpha=0.5 ,we get train AUC of 0.78 and Test AUC of 0.70

d) Confusion Matrix

Train data

In [152]:

```
# we will pick a threshold that will give the least fpr
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.rou
nd(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
```

In [153]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.5098205816664191 for threshold 0.433
Train confusion matrix
[[ 7767  3316]
 [16927 45186]]
```

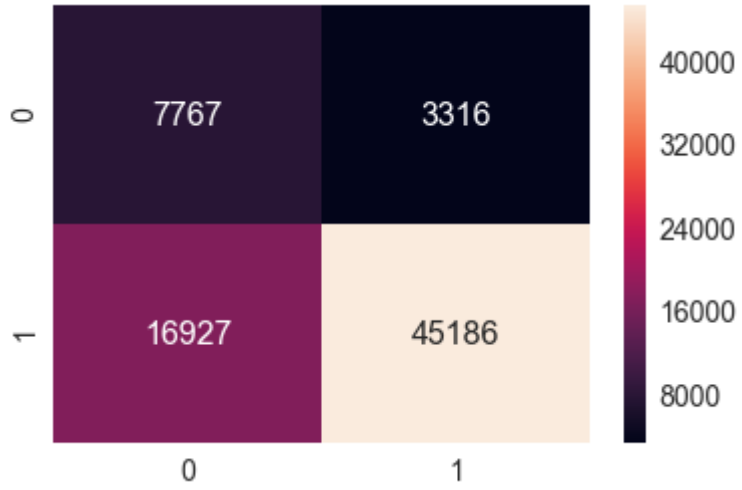
In [154]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(
y_train_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[154]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f4d081bdd8>
```



Summary on Train data

In the following confusion matrix we observe that the model has 44k true positives while false positives are only 3k

It has large number of false negatives which are close to 17k

Test data

In [155]:

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

```
Test confusion matrix
[[ 3298  2161]
 [ 8855 21738]]
```

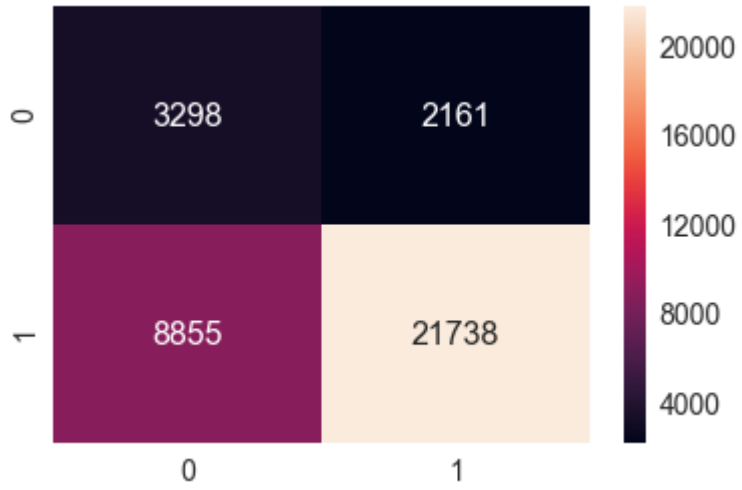
In [156]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_
test_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[156]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f48ed2fb38>
```



In [ ]:

```

```

Summary on Test data

1. The number of true positives dominate ,there are 21k in number,
2. The least number among 4 quantites is false positive which are 2077
3. similar trend is observed for false negatives which are roughly 9k

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

In [157]:

```
# Please write all the code with proper documentation

X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_tf
idf, train_title_tfidf, train_school_state_category_one_hot,train_teacher_prefix_catego
ries_one_hot, previously_posted_projects_normalized_train, train_project_grade_category
_one_hot, price_normalized_train)).tocsr()

X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_tfidf,
test_title_tfidf, test_school_state_category_one_hot, test_teacher_prefix_categories_on
e_hot, previously_posted_projects_normalized_test, test_project_grade_category_one_hot,
price_normalized_test)).tocsr()
print(X_train.shape)

print(X_test.shape)
```

```
(73196, 16747)
(36052, 16747)
```

A) random alpha values

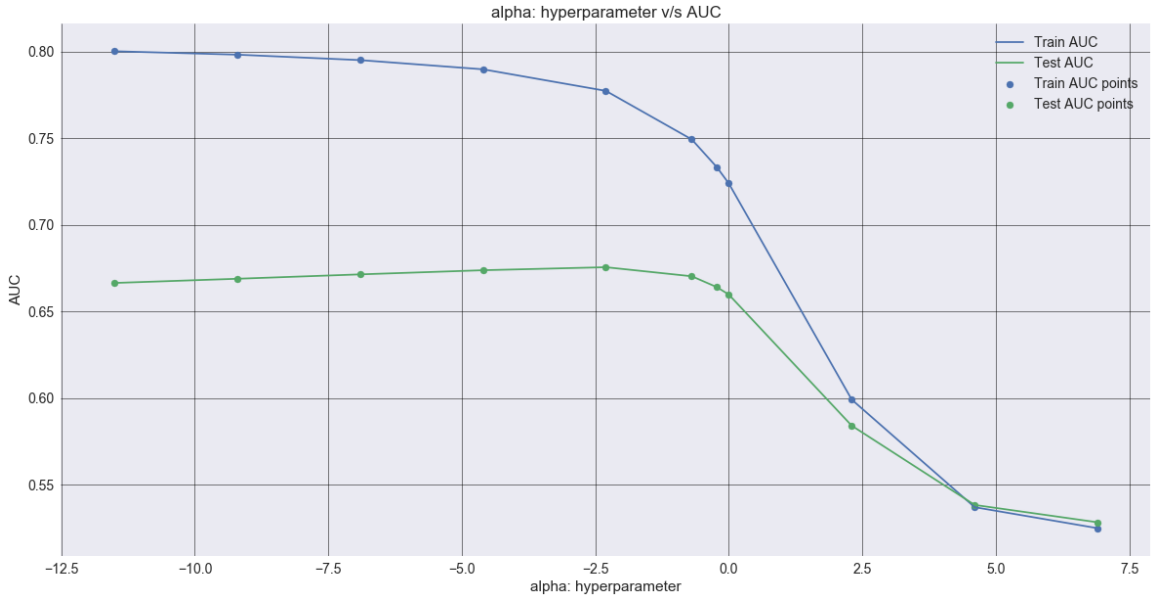
In [158]:

```
train_auc = []
test_auc = []
log_alphas =[]

alphas = [0.00001, 0.0001, 0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior=[0.5,0.5])
    nb.fit(X_train, y_train)

    y_train_pred = batch_predict(nb, X_train)
    y_test_pred = batch_predict(nb, X_test)


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    test_auc.append(roc_auc_score(y_test, y_test_pred))

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

```
100%|██████████| 11/11 [00:05<00:00,  1.89it/s]
100%|██████████| 11/11 [00:00<00:00, 5522.12it/s]
```

In [159]:

```
plt.figure(figsize=(20,10))

plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, test_auc, label='Test AUC')

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, test_auc, label='Test AUC points')

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



b) GridSearch-cv using cv=10 (K fold cross validation)

In [160]:

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

parameters = {'alpha':[0.00001, 0.0001, 0.001, 0.01, 0.1,0.25,0.5,0.8, 1,100]}

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

clf.fit(X_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
test_auc = clf.cv_results_['mean_test_score']
test_auc_std= clf.cv_results_['std_test_score']
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
[CV] alpha=1e-05 .....................................................
[CV] ..................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.4s remaining:
0.0s
```

```
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
```

```
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
```

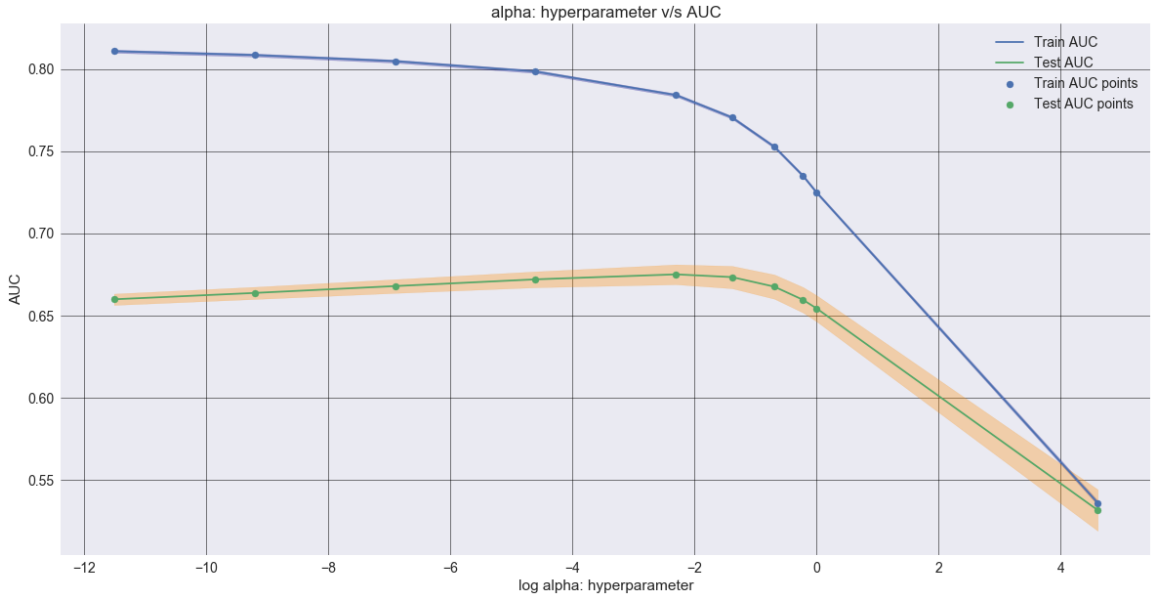
```
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.2s
```

```
[CV] alpha=100 .......................................................
[CV] ..................................... alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ..................................... alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ..................................... alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ..................................... alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ..................................... alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ..................................... alpha=100, total=   0.2s
[CV] alpha=100 .......................................................
[CV] ..................................... alpha=100, total=   0.2s

[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:   40.4s finished
```

In [161]:

```
alphas = [0.00001, 0.0001, 0.001, 0.01, 0.1,0.25,0.5,0.8, 1,100]
log_alphas =[]

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

plt.figure(figsize=(20,10))

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,a
lpha=0.3,color='darkblue')

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

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, test_auc, label='Test 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%|██████████████████████████████████████████████████████████████████████████████| 10/10 [00:00<?, ?it/s]
```



In [ ]:

c) Train model using the best hyperparameter value of alpha

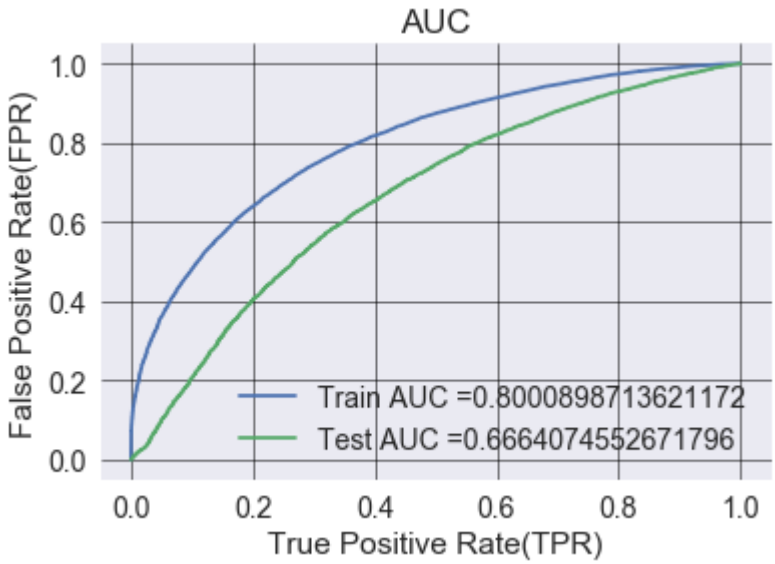
In [162]:

```
#https://datascience.stackexchange.com/questions/21877/how-to-use-the-output-of-gridsea
rch
#choosing the best hyperparameter

best_alpha2=clf.best_params_
print(best_alpha2)
```

```
{'alpha': 0.1}
```

In [163]:

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

nb_tfidf.fit(X_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
he positive class
# not the predicted outputs

y_train_pred = batch_predict(nb_tfidf, X_train)
y_test_pred = batch_predict(nb_tfidf, X_test)

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()
```



In [ ]:

```

```

Summary

From given plot we observe that at alpha=0.1 we get train AUC of 0.80 and test AUC of 0.66

d) Confusion Matrix

Train data

In [164]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.5242475287699654 for threshold 0.509
Train confusion matrix
[[ 8093  2990]
 [17520 44593]]
```

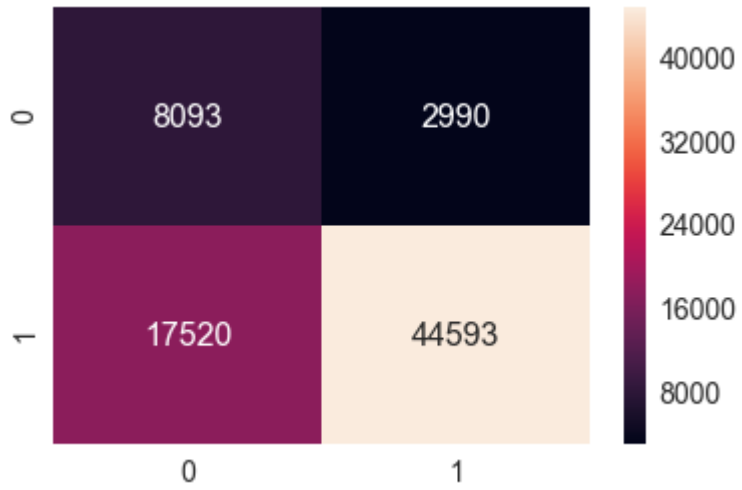
In [165]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(
y_train_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[165]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f494b8cb38>
```



Summary for train data

1. For training data we get roughly 44k true positives
2. Again we have roughly 18k false negatives which are alot in number

Test data

In [166]:

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

```
Test confusion matrix
[[ 3051  2408]
 [ 9365 21228]]
```

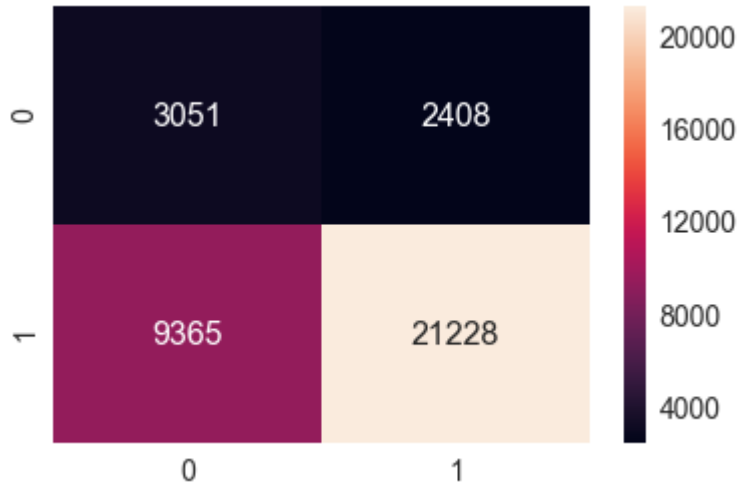
In [167]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_
test_pred, best_t)), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[167]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1f48dd897b8>
```



Summary for test data

1. we have roughly 21000 true positives for test data and roughly 2400 false positives
2. Again false negatives are pretty high in number(9k)

Select best 20 features of both Positive and negative class for both the sets of data

Set1 : BOW

In [191]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_bo
w, train_title_bow, train_school_state_category_one_hot,train_teacher_prefix_categories
_one_hot, previously_posted_projects_normalized_train, train_project_grade_category_one
_hot, price_normalized_train)).tocsr()

X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_bow, t
est_title_bow, test_school_state_category_one_hot, test_teacher_prefix_categories_one_h
ot, previously_posted_projects_normalized_test, test_project_grade_category_one_hot, pr
ice_normalized_test)).tocsr()

print(X_train.shape)

print(X_test.shape)
```

```
(73196, 16747)
(36052, 16747)
```

In [192]:

```
bow_features_names = []


for feature in vectorizer_cat.get_feature_names() :
    bow_features_names.append(feature)

for feature in vectorizer_subcat.get_feature_names() :
    bow_features_names.append(feature)

for feature in vectorizer_school_state.get_feature_names() :
    bow_features_names.append(feature)

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

for feature in vectorizer_prefix.get_feature_names() :
    bow_features_names.append(feature)
```

In [193]:

```
for feature in vectorizer_bow_title.get_feature_names() :
    bow_features_names.append(feature)
```

In [194]:

```
for feature in vectorizer_bow_essay.get_feature_names() :
    bow_features_names.append(feature)
```

In [195]:

```
bow_features_names.append("price")

bow_features_names.append("teacher_number_of_previously_posted_projects")
```

In [196]:

```
len(bow_features_names)
```

Out[196]:

```
16745
```

In [197]:

```
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()   #class 0
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()   #class1
```

In [198]:

```
print(neg_class_prob_sorted[-20:],pos_class_prob_sorted[-20:])
```

```
[14089  2374  3297 13461 11532  7781 10183   290  7560  2556 14004  8355
  7714  5999  7290  8492  2387  7294 11072 12205] [ 1604 12593 14045  2374
2556  3297   290  7560 13461 10183 14004  8355
  7714  5999  7290  8492  2387  7294 11072 12205]
```

In [199]:

```
print('Top 20 features from negative class:')
print(np.take(bow_features_names, neg_class_prob_sorted[-20:]))
print('-*'*50)
print('Top 20 features from positive class:')
print(np.take(bow_features_names, pos_class_prob_sorted[-20:]))
```

```
Top 20 features from negative class:
['sir' 'therapy' 'alphabet' 'rotations' 'palsy' 'folding' 'lungs' 'begins'
 'felt' 'whiteboard' 'shown' 'groundbreaking' 'flex' 'defying' 'exhibited'
 'hardwood' 'thrive' 'exhibits' 'norm' 'preferably']
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
-*-*-*-*-*-*-*-*-*-*-*-*
Top 20 features from positive class:
['motivated' 'push' 'silence' 'therapy' 'whiteboard' 'alphabet' 'begins'
 'felt' 'rotations' 'lungs' 'shown' 'groundbreaking' 'flex' 'defying'
 'exhibited' 'hardwood' 'thrive' 'exhibits' 'norm' 'preferably']
```

In [ ]:

```

```

Set2: TF-IDF

In [200]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

# with the same hstack function we are concatinating a sparse matrix and a dense matirx
:)
X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_essay_tf
idf, train_title_tfidf, train_school_state_category_one_hot,train_teacher_prefix_catego
ries_one_hot, previously_posted_projects_normalized_train, train_project_grade_category
_one_hot, price_normalized_train)).tocsr()

X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_essay_tfidf,
test_title_tfidf, test_school_state_category_one_hot, test_teacher_prefix_categories_on
e_hot, previously_posted_projects_normalized_test, test_project_grade_category_one_hot,
price_normalized_test)).tocsr()

print(X_train.shape)

print(X_test.shape)
```

```
(73196, 16747)
(36052, 16747)
```

In [201]:

```
tfidf_features_names = []
```

In [202]:

```
for feature in vectorizer_cat.get_feature_names() :
    tfidf_features_names.append(feature)

for feature in vectorizer_subcat.get_feature_names() :
    tfidf_features_names.append(feature)

for feature in vectorizer_school_state.get_feature_names() :
    tfidf_features_names.append(feature)

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

for feature in vectorizer_prefix.get_feature_names() :
    tfidf_features_names.append(feature)
```

In [203]:

```
for feature in vectorizer_tfidf_title.get_feature_names() :
    tfidf_features_names.append(feature)

for feature in vectorizer_tfidf_essay.get_feature_names() :
    tfidf_features_names.append(feature)
```

In [204]:

```
tfidf_features_names.append('price')
tfidf_features_names.append('teacher_number_of_previously_posted_projects')
```

In [205]:

```
len(tfidf_features_names)
```

Out[205]:

```
16745
```

In [ ]:

```

```

In [206]:

```
neg_class_prob_sorted = nb_tfidf.feature_log_prob_[0, :].argsort()   #class 0
pos_class_prob_sorted = nb_tfidf.feature_log_prob_[1, :].argsort()   #class1
```

In [207]:

```
print(neg_class_prob_sorted[-20:],pos_class_prob_sorted[-20:])
```

```
[16694    37 16728    24     6 16737     9     0 16689     2 12205    35
     7    27    26    28 16739     5     4 16738] [   37 16728 16719     6
9    24 16737     0     7    35     2 12205
 16689    27    28    26 16739     5     4 16738]
```

In [208]:

```
print('Top 20 features from negative class:')
print(np.take(tfidf_features_names, neg_class_prob_sorted[-10:]))
print('-*'*50)
print('Top 10 features from positive class:')
print(np.take(tfidf_features_names, pos_class_prob_sorted[-10:]))
```

```
Top 20 features from negative class:
['preferably' 'SpecialNeeds' 'SpecialNeeds' 'Literature_Writing'
 'Literacy' 'Mathematics' 'zoom' 'Math_Science' 'Literacy_Language'
 'zoology']
-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
-*-*-*-*-*-*-*-*-*-*-*-*
Top 10 features from positive class:
['Health_Sports' 'preferably' 'years' 'Literature_Writing' 'Mathematics'
 'Literacy' 'zoom' 'Math_Science' 'Literacy_Language' 'zoology']
```

Conclusions

In [209]:

```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prett
ytable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha:Hyper Parameter", " Test AUC"]

x.add_row(["BOW", "Naive Bayes", 0.5, 0.70])
x.add_row(["TFIDF", "Naive Bayes", 0.1, 0.66])

print(x)
```

```
+------------+-------------+-----------------------+-----------+
| Vectorizer |    Model    | Alpha:Hyper Parameter |  Test AUC |
+------------+-------------+-----------------------+-----------+
|    BOW     | Naive Bayes |          0.5          |    0.7    |
|   TFIDF    | Naive Bayes |          0.1          |    0.66   |
+------------+-------------+-----------------------+-----------+
```