DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [3]:

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

```
C:\Users\ADMIN\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; al
iasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [4]:

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

In [5]:

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

```
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']
```

In [6]:

```
print("Number of data points in resource data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in resource data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[6]:

	id	description	quantity	price
		LC652 - Lakeshore Double-Space Mobile Drying		

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [7]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
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", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "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:"Math &
Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        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]))
```

1.3 preprocessing of `project_subject_subcategories`

In [8]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

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

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", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "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:"Math &
Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        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)
```

```
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]))
```

1.3 Text preprocessing

In [9]:

```
# 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 [10]:

```
project_data.head(2)
```

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade

In [11]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

2. Naive Bayes

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

Sampling of data

I took a sample of 50000 data points and stored it in sample_data

In [12]:

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

```
    # d. Y-axis label

sample_data=project_data.sample(50000)
sample_data.shape
```

Out[12]:

```
(50000, 20)
```

Splitting the data

The sample data is splitted into 3 parts , train data, test data, and crossvalidation data.

In [13]:

```
from sklearn.model_selection import train_test_split
data=sample_data
data.head()
y=data.project_is_approved
x=data.drop('project_is_approved',axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,test_size=0.2,stratify=y)
x_train,x_cv,y_train,y_cv=train_test_split(x_train,y_train,train_size=0.8,test_size=0.2,stratify=y
_train)


print("shape of train data ")
print(x_train.shape)
print(y_train.shape)
print("shape of test data ")
print(x_test.shape)
print(y_test.shape)
print("shape of crossvalidation data ")
print(x_cv.shape)
print(y_cv.shape)
```

```
shape of train data
(32000, 19)
(32000,)
shape of test data
(10000, 19)
(10000,)
shape of crossvalidation data
(8000, 19)
(8000,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [14]:

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

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

Out[14]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'clean_subcategories', 'essay', 'price', 'quantity'],
      dtype='object')
```

Vectorizing numerical and categorical data

In [74]:

```
def veccat(x):
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.preprocessing import Normalizer
    from scipy.sparse import hstack
    vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=T
rue)
    vectorizer.fit(x['clean_categories'].values)
    print(vectorizer.get_feature_names())



    categories_one_hot = vectorizer.transform(x['clean_categories'].values)
    print("Shape of matrix after one hot encodig ",categories_one_hot.shape)

    vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
binary=True)
    vectorizer.fit(x['clean_subcategories'].values)
    print(vectorizer.get_feature_names())



    subcategories_one_hot = vectorizer.transform(x['clean_subcategories'].values)
    print("Shape of matrix after one hot encodig ",subcategories_one_hot.shape)


    vectorizer = CountVectorizer(lowercase=False, binary=True)
    vectorizer.fit(x['school_state'].values)
    print(vectorizer.get_feature_names())



    state_one_hot = vectorizer.transform(x['school_state'].values)
    print("Shape of matrix after one hot encodig ",state_one_hot.shape)

    x = x.replace(np.nan, '', regex=True)
    vectorizer = CountVectorizer( vocabulary=['Mrs.', 'Ms.','Mr.','Teacher','Dr.'],lowercase=False)
    vectorizer.fit(x['teacher_prefix'].values)
    print(vectorizer.get_feature_names())



    prefix_one_hot = vectorizer.transform(x['teacher_prefix'].values)
    print("Shape of matrix after one hot encodig ",prefix_one_hot.shape)

    vocab=[]
    for i in x['project_grade_category'].values:
        vocab.append(i)
    v_set=set(vocab)
    vocab=list(v_set)
    vectorizer = CountVectorizer(vocabulary=vocab,lowercase=False)
    vectorizer.fit(x['project_grade_category'].values)




    grade_one_hot = vectorizer.transform(x['project_grade_category'].values)
    print("Shape of matrix after one hot encodig ",grade_one_hot.shape)

    price_scalar =  Normalizer(copy=False,norm='l2')
    price_scalar.fit(x['price'].values.reshape(1,-1)) # finding the mean and standard deviation of
this data



    # Now standardize the data with above maen and variance.
    price_standardized = price_scalar.transform(x['price'].values.reshape(1, -1))
    price_standardized=np.transpose(price_standardized)

    projects_scalar = Normalizer(copy=False,norm='l2')
    projects_scalar.fit(x['teacher_number_of_previously_posted_projects'].values.reshape(1,-1)) # f
```

```
    projects_scalar.fit(x['teacher_number_of_previously_posted_projects'].values.reshape(1,-1)) # f
inding the mean and standard deviation of this data

    # Now standardize the data with above maen and variance.
    projects_standardized =
projects_scalar.transform(x['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))
    projects_standardized =np.transpose(projects_standardized)

    qty_scalar= Normalizer(copy=False,norm='12')
    qty_scalar.fit(x['quantity'].values.reshape(1,-1)) # finding the mean and standard deviation of
this data


    # Now standardize the data with above maen and variance.
    qty_standardized = qty_scalar.transform(x['quantity'].values.reshape(1, -1))
    qty_standardized=np.transpose(qty_standardized)

    X1 = hstack((categories_one_hot, subcategories_one_hot,state_one_hot,prefix_one_hot,grade_one_h
ot, price_standardized,projects_standardized,qty_standardized))
    print(X1.shape)
    return(X1)
```

In [75]:

```
x=veccat(x_train)
t=veccat(x_test)
cv=veccat(x_cv)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (32000, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (32000, 30)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encodig  (32000, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (32000, 5)
Shape of matrix after one hot encodig  (32000, 4)
(32000, 102)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (10000, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (10000, 30)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encodig  (10000, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (10000, 5)
Shape of matrix after one hot encodig  (10000, 4)
(10000, 102)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (8000, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (8000, 30)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encodig  (8000, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (8000, 5)
Shape of matrix after one hot encodig  (8000, 4)
(8000, 102)
```

In [76]:

```
# For finding the feature list of categorical and numerical data
def features(x):
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.preprocessing import MinMaxScaler
    from scipy.sparse import hstack
    vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=T
rue)
    vectorizer.fit(x['clean_categories'].values)

    feature_list.extend(vectorizer.get_feature_names())
    vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
binary=True)
    vectorizer.fit(x['clean_subcategories'].values)

    feature_list.extend(vectorizer.get_feature_names())
    vectorizer = CountVectorizer(lowercase=False, binary=True)
    vectorizer.fit(x['school_state'].values)

    feature_list.extend(vectorizer.get_feature_names())
    x = x.replace(np.nan, '', regex=True)
    vectorizer = CountVectorizer( vocabulary=['Mrs.', 'Ms.','Mr.','Teacher','Dr.'],lowercase=False)
    vectorizer.fit(x['teacher_prefix'].values)

    feature_list.extend(vectorizer.get_feature_names())
    vocab=[]
    for i in x['project_grade_category'].values:
        vocab.append(i)
    v_set=set(vocab)
    vocab=list(v_set)
    vectorizer = CountVectorizer(vocabulary=vocab,lowercase=False)
    vectorizer.fit(x['project_grade_category'].values)
    feature_list.extend(vectorizer.get_feature_names())
    feature_list.append("price")
    feature_list.append("teacher_number_of_previously_posted_projects")
    feature_list.append("quantity")
```

In [77]:

```
feature_list=[]
features(x_train)
print(len(feature_list))
```

```
102
```

2.3 Make Data Model Ready: encoding eassay, and project_title

Vectorizing essay and project_title

In [19]:

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

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
```

```
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
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 [20]:

```
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', 'e
ach', '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', "d
esn'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 [21]:

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

In [78]:

```
train_essay=[]
test_essay=[]
cv_essay=[]
train_title=[]
test_title=[]
cv_title=[]
train_essay=preprocessing(x_train['essay'])
test_essay=preprocessing(x_test['essay'])
cv_essay=preprocessing(x_cv['essay'])

train_title=preprocessing(x_train['project_title'])
test_title=preprocessing(x_test['project_title'])
cv_title=preprocessing(x_cv['project_title'])
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|████████████████████████████████████| 32000/32000 [00:33<00:00, 960.93it/s]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|████████████████████████████████████| 10000/10000 [00:09<00:00, 1021.39it/s]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|████████████████████████████████████| 8000/8000 [00:08<00:00, 929.10it/s]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|████████████████████████████████████| 32000/32000 [00:01<00:00, 23441.88it/s]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|████████████████████████████████████| 10000/10000 [00:00<00:00, 24211.67it/s]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|████████████████████████████████████| 8000/8000 [00:00<00:00, 24463.43it/s]
```

Encoding using BOW

In [79]:

```
feature_bow = feature_list.copy()
feature_tfidf = feature_list.copy()
```

In [24]:

```
from scipy.sparse import hstack
from sklearn.preprocessing import StandardScaler
def bow_text(train_essay, test_essay, cv_essay,train_title,test_title,cv_title,x1,t1,cv1):
    from scipy.sparse import hstack

    vectorizer = CountVectorizer()

    vectorizer.fit(train_essay)

    text_bow=vectorizer.transform(train_essay)

    text_bow1 = vectorizer.transform(test_essay)

    text_bow2 = vectorizer.transform(cv_essay)
    feature_bow.extend(vectorizer.get_feature_names())
    vectorizer.fit(train_title)
    title_bow=vectorizer.transform(train_title)

    title_bow1 = vectorizer.transform(test_title)

    title_bow2 = vectorizer.transform(cv_title)

    x1 = hstack((x1,text_bow,title_bow )).tocsr()
    t1= hstack((t1,text_bow1,title_bow1 )).tocsr()
    cv1 = hstack((cv1,text_bow2,title_bow2 )).tocsr()
```

```
    feature_bow.extend(vectorizer.get_feature_names())


    return x1,t1,cv1
```

In [80]:

```
# Data matrix using BOW
x1,t1,cv1=bow_text(train_essay, test_essay, cv_essay,train_title,test_title,cv_title,x,t,cv)
print(x1.shape)
print(t1.shape)
print(cv1.shape)
```

```
(32000, 44243)
(10000, 44243)
(8000, 44243)
```

In [81]:

```
# Feature list for BOW encoding
print(len(feature_bow))
```

```
44243
```

Encoding using TFIDF

In [27]:

```
def tfidf_text(train_essay, test_essay, cv_essay,train_title,test_title,cv_title,x1,t1,cv1):
    from sklearn.feature_extraction.text import TfidfVectorizer
    vectorizer = TfidfVectorizer()
    vectorizer.fit(train_essay)
    feature_tfidf.extend(vectorizer.get_feature_names())
    text_tfidf=vectorizer.transform(train_essay)
    text_tfidf1 = vectorizer.transform(test_essay)
    text_tfidf2 = vectorizer.transform(cv_essay)
    vectorizer.fit(train_title)
    feature_tfidf.extend(vectorizer.get_feature_names())
    title_tfidf=vectorizer.transform(train_title)
    title_tfidf1 = vectorizer.transform(test_title)
    title_tfidf2 = vectorizer.transform(cv_title)
    x1 = hstack((x1,text_tfidf,title_tfidf ))
    t1= hstack((t1,text_tfidf1,title_tfidf1 ))
    cv1 = hstack((cv1,text_tfidf2,title_tfidf2 ))



    return x1,t1,cv1
```

In [82]:

```
x2,t2,cv2=tfidf_text(train_essay, test_essay, cv_essay,train_title,test_title,cv_title,x,t,cv)
print(x2.shape)
print(t2.shape)
print(cv2.shape)
```

```
(32000, 44243)
(10000, 44243)
(8000, 44243)
```

In [83]:

```
# Feature list for TFIDF encoding
print(len(feature_tfidf))
```

```
44243
```

2.4 Appling NB() on different kind of featurization as mentioned in the instructions

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

In [30]:

```
# Function for finding the optimal alpha in BOW and TFIDF
def alpha_from_roc(x_train,y_train,x_cv,y_cv):
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.model_selection import cross_val_score
    from sklearn.metrics import roc_curve
    from sklearn.metrics import roc_auc_score
    from sklearn.preprocessing import MinMaxScaler

    a_range=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,7,9,10,20,35,45,50,100]
    scores_auc=[]

    for a in a_range:
        clf=MultinomialNB(alpha=a)
        clf.fit(x_train,y_train)
        prob= clf.predict_proba(x_cv)
        prob=prob[:,1]
        scores_auc.append(roc_auc_score(y_cv,prob))
    # Chosen optimal k with the highest AUC score
    optimal_alpha = a_range[scores_auc.index(max(scores_auc))]

    clf = MultinomialNB(alpha=optimal_alpha)
    clf.fit(x_train,y_train)
    # predict probabilities
    prob = clf.predict_proba(x_cv)
    # keep probabilities for the positive outcome only
    prob = prob[:, 1]
    prob1=clf.predict_proba(x_train)
    prob1=prob1[:, 1]
    # calculate roc curve
    fpr, tpr, thresholds = roc_curve(y_cv, prob)
    fpr1, tpr1, thresholds1 = roc_curve(y_train, prob1)
    # plot no skill
    plt.plot([0, 1], [0, 1], linestyle='--')
    # plot the roc curve for the model
    plt.plot(fpr, tpr, marker='.',label="CrossValidated or test ROC")
    plt.plot(fpr1, tpr1, marker='.' , label='Train ROC')
    plt.legend()
    # show the plot
    plt.title("Receiver Operating Characteristics")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.show()
    print("Optimal alpha ",optimal_alpha)
    print("AUC: ", max(scores_auc))
    plt.plot(np.log10(a_range), scores_auc, label='CV or TEST AUC')
    plt.scatter(np.log10(a_range), scores_auc, label='CV or TEST AUC points')
    plt.xlabel("LOG(alpha): hyperparameter")
    plt.ylabel("AUC")
    plt.title("ERROR PLOTS")
    plt.legend()

    plt.show()
```

In [31]:

```
# K- Fold cross validation
def kfold_cv(x_train,y_train):
    from sklearn.model_selection import  cross_val_score
    from sklearn.naive_bayes import MultinomialNB
    a_range=[0.1,0.5,1,10,20,35,45,50,100]
    scores_cv=[]
    for a in a_range:
        clf=MultinomialNB(alpha=a)
        scores_cv.append(np.mean(cross_val_score(clf, x_train, y_train, cv=3, n_jobs=1)))
```

```
    for i in range(len(scores_cv)):
        if scores_cv[i]==max(scores_cv):
            print("Optimal alpha :",a_range[i])
            print("CV SCORE ",max(scores_cv))
```

In [32]:

```
# Gridsearchcv function
def grid_search(x_train,y_train,x_cv,y_cv):
    from sklearn.naive_bayes import  MultinomialNB
    from sklearn.model_selection import GridSearchCV
    from sklearn.metrics import roc_curve
    from sklearn.metrics import roc_auc_score
    from sklearn import preprocessing

    a_range=[0.1,0.5,1,10,20,35,45,50,100]
    nb=MultinomialNB(alpha=a_range)
    parameters = dict(alpha=a_range)
    # Fitted the model on train data
    clf = GridSearchCV(nb, parameters, cv=3,verbose=15, scoring='roc_auc')
    clf.fit(x_train, y_train)
    train_auc= clf.cv_results_['mean_train_score']
    cv_auc = clf.cv_results_['mean_test_score']
    max_score= clf.best_score_

    opt_par=clf.best_params_
    print("Optimal alpha ",opt_par)
    print("AUC: ", max_score)
    plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
    plt.plot(parameters['alpha'], train_auc, label='TRAIN AUC')
    plt.xlabel("Alpha: hyperparameter")
    plt.ylabel("AUC")
    plt.title("ERROR PLOTS")
    plt.legend()
    plt.show()
    # Found out the score for crossvalidated data
    print("Accuracy on crossvalidated data: " , clf.score(x_cv,y_cv))
```

In [33]:

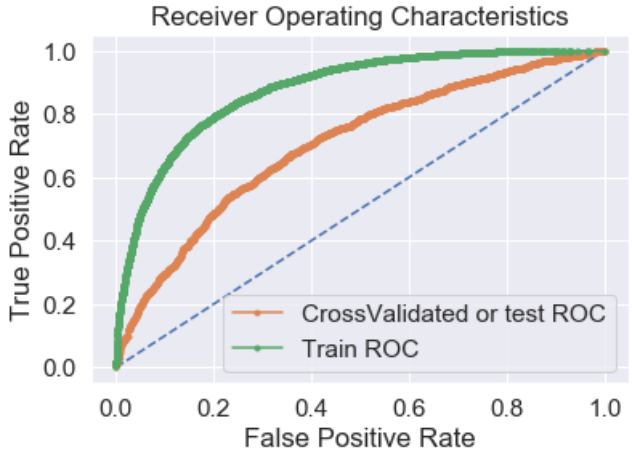
```
# Predictions  on test data in BOW and TFIDF

def predict(x_train,y_train,x_test,y_test,a):
    from sklearn.naive_bayes import  MultinomialNB
    clf=MultinomialNB(alpha=a)
    clf.fit(x_train,y_train)
    prob1=clf.predict_proba(x_train)
    prob1=prob1[:,1]
    fpr, tpr, threshould = roc_curve(y_train, prob1)
    t = threshould[np.argmax(tpr*(1-fpr))]
    prob= clf.predict_proba(x_test)
    prob=prob[:,1]


    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    print("train AUC =",str(auc(fpr, tpr)))
    predictions = []
    for i in prob:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

2.4.1 Applying Naive Bayes on BOW, SET 1

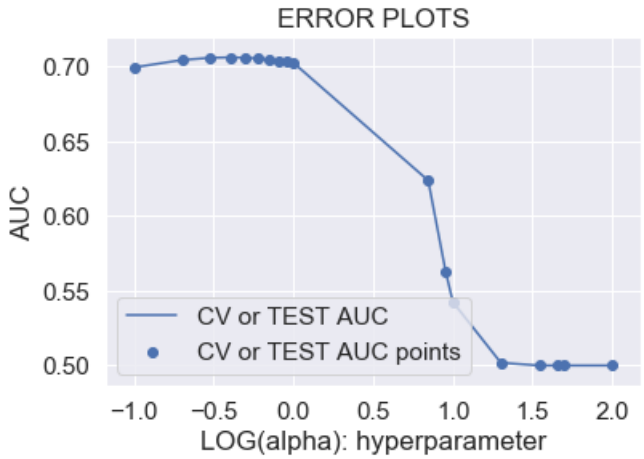
ROC SCORE on CV data

In [84]:

```
# Please write all the code with proper documentation
alpha_from_roc(x1,y_train,cv1,y_cv)
```




```
Optimal alpha  0.4
AUC:  0.7060126713119443
```



K FOLD Cross Validation

In [85]:

```
kfold_cv(x1,y_train)
```

```
Optimal alpha : 20
CV SCORE  0.849093751658438
Optimal alpha : 35
CV SCORE  0.849093751658438
Optimal alpha : 45
CV SCORE  0.849093751658438
Optimal alpha : 50
CV SCORE  0.849093751658438
Optimal alpha : 100
CV SCORE  0.849093751658438
```

GridsearchCV

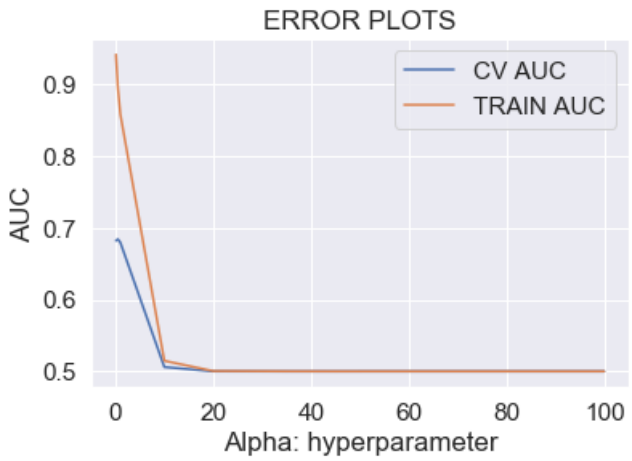
In [86]:

```
grid_search(x1,y_train,cv1,y_cv)
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[CV] alpha=0.1 .......................................................
[CV] .............. alpha=0.1, score=0.6767896147038391, total=   0.0s

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

[CV] alpha=0.1 .......................................................
[CV] .............. alpha=0.1, score=0.6775417867652556, total=   0.0s

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.2s remaining:    0.0s

[CV] alpha=0.1 .......................................................
[CV] .............. alpha=0.1, score=0.6918757337772314, total=   0.0s

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.3s remaining:    0.0s

[CV] alpha=0.5 .......................................................
[CV] .............. alpha=0.5, score=0.6762265486288702, total=   0.0s

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.4s remaining:    0.0s

[CV] alpha=0.5 .......................................................
[CV] .............. alpha=0.5, score=0.6791153611667171, total=   0.0s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.5s remaining:    0.0s

[CV] alpha=0.5 .......................................................
[CV] .............. alpha=0.5, score=0.6968866058090899, total=   0.0s

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.7s remaining:    0.0s

[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.6698392581970501, total=   0.0s

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.8s remaining:    0.0s

[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.675094038652372, total=   0.0s

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    1.0s remaining:    0.0s

[CV] alpha=1 .........................................................
[CV] ................ alpha=1, score=0.6936274323113342, total=   0.0s

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.2s remaining:    0.0s

[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.5081994504096553, total=   0.0s

[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    1.3s remaining:    0.0s

[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.5026057879118927, total=   0.0s

[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    1.4s remaining:    0.0s

[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.5071767693496743, total=   0.0s

[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    1.5s remaining:    0.0s
```

```
[CV] alpha=20 .........................................................
[CV] ............... alpha=20, score=0.4999447940819256, total=   0.0s

[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    1.7s remaining:    0.0s

[CV] alpha=20 .........................................................
[CV] ............... alpha=20, score=0.4999447940819256, total=   0.0s

[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    1.8s remaining:    0.0s

[CV] alpha=20 .........................................................
[CV] ............... alpha=20, score=0.5002555117911126, total=   0.0s
[CV] alpha=35 .........................................................
[CV] ............................. alpha=35, score=0.5, total=   0.0s
[CV] alpha=35 .........................................................
[CV] ............... alpha=35, score=0.4999447940819256, total=   0.0s
[CV] alpha=35 .........................................................
[CV] ............... alpha=35, score=0.4999447940819256, total=   0.0s
[CV] alpha=45 .........................................................
[CV] ............................. alpha=45, score=0.5, total=   0.0s
[CV] alpha=45 .........................................................
[CV] ............................. alpha=45, score=0.5, total=   0.0s
[CV] alpha=45 .........................................................
[CV] ............... alpha=45, score=0.4999447940819256, total=   0.0s
[CV] alpha=50 .........................................................
[CV] ............................. alpha=50, score=0.5, total=   0.0s
[CV] alpha=50 .........................................................
[CV] ............................. alpha=50, score=0.5, total=   0.0s
[CV] alpha=50 .........................................................
[CV] ............... alpha=50, score=0.4999447940819256, total=   0.0s
[CV] alpha=100 ........................................................
[CV] ............................ alpha=100, score=0.5, total=   0.0s
[CV] alpha=100 ........................................................
[CV] ............................ alpha=100, score=0.5, total=   0.0s
[CV] alpha=100 ........................................................
[CV] ............................ alpha=100, score=0.5, total=   0.0s

[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    3.5s finished

Optimal alpha  {'alpha': 0.5}
AUC:  0.684075771542165
```

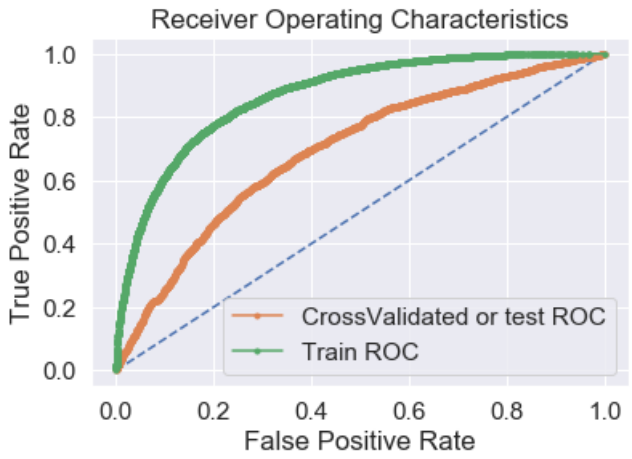



```
Accuracy on crossvalidated data:  0.7057318495536917
```

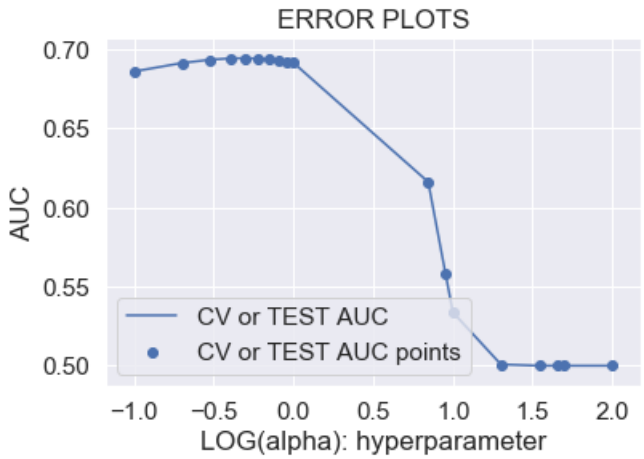
ROC Score on Test Data

In [87]:

```
alpha_from_roc(x1,y_train,t1,y_test)
```




```
Optimal alpha  0.5
AUC:  0.694527257996402
```



Confusion Matrix

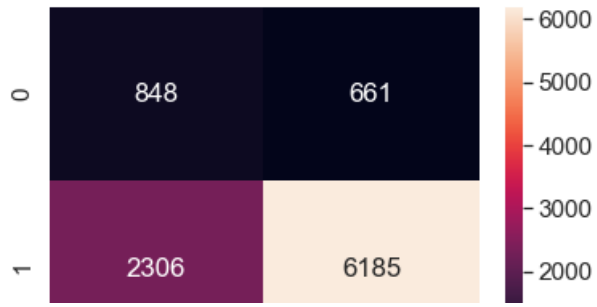
In [88]:

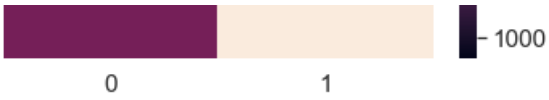
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import  MultinomialNB
labels=[0,1]
print("Test confusion matrix")
cm=confusion_matrix(y_test, predict(x1, y_train, t1, y_test,0.5),labels)
sns.set(font_scale=1.4)
sns.heatmap(cm,fmt='d',annot=True)
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.6197890340537403 for threshold 0.958
train AUC = 0.8697639499814185
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x6254bc88>
```





2.4.1.1 Top 10 important features of positive class from SET 1

In [89]:

```
# Please write all the code with proper documentation
# alpha is selected based on k-fold CV score
from sklearn.naive_bayes import  MultinomialNB
clf = MultinomialNB(alpha=20)
clf.fit(x1,y_train)
important_feature_pos = pd.DataFrame(
    {'features': feature_bow,
     'prob_value': clf.feature_log_prob_[0],
    })

top_10_pos=important_feature_pos.sort_values(['prob_value'], ascending=[True]).tail(10)
print("Top 10  features for positive class")
print(top_10_pos)
```

```
Top 10  features for positive class
        features  prob_value
20830       need   -5.944460
19079       many   -5.865534
20665     nannan   -5.814377
21212        not   -5.642073
14682       help   -5.638807
17983      learn   -5.604447
6226   classroom   -5.433094
17995   learning   -5.251436
27160     school   -4.962832
29877   students   -3.857914
```

2.4.1.2 Top 10 important features of negative class from SET 1

In [90]:

```
# Please write all the code with proper documentation
important_feature_neg = pd.DataFrame(
    {'features': feature_bow,
     'prob_value': clf.feature_log_prob_[1],
    })

top_10_neg=important_feature_neg.sort_values(['prob_value'], ascending=[True]).tail(10)
print("Top 10  features for negative class")
print(top_10_neg)
```

```
Top 10  features for negative class
        features  prob_value
25144    reading   -5.334478
20665     nannan   -5.232575
19079       many   -5.226053
14682       help   -5.076711
17983      learn   -5.046252
21212        not   -5.000603
6226   classroom   -4.734814
17995   learning   -4.705145
27160     school   -4.349661
29877   students   -3.196598
```

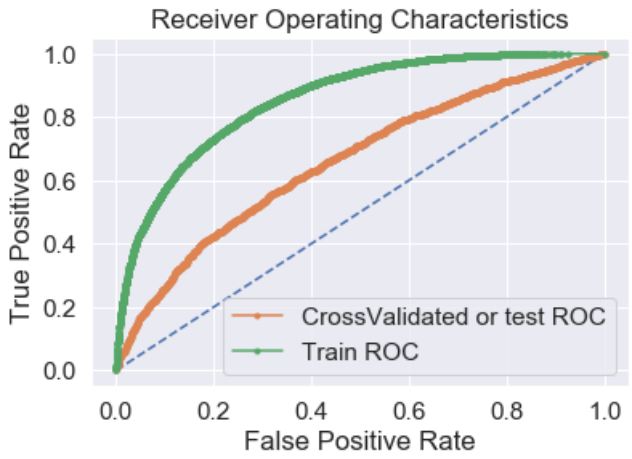
2.4.2 Applying Naive Bayes on TFIDF, SET 2

ROC SCORE on CV data

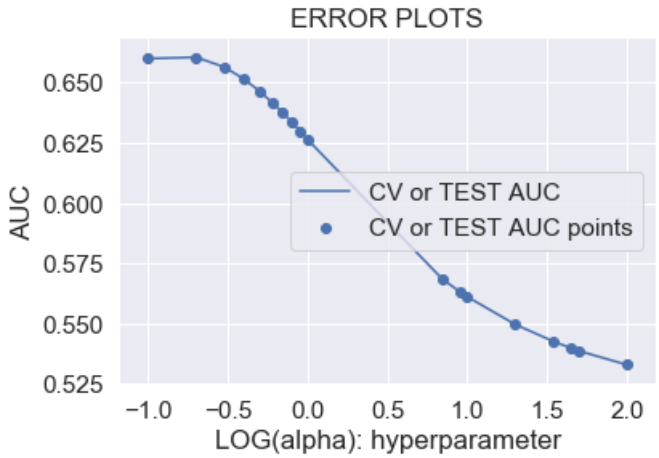
In [91]:

In [91]:

```
# Please write all the code with proper documentation
alpha_from_roc(x2,y_train,cv2,y_cv)
```




```
Optimal alpha  0.2
AUC:  0.6605709542366032
```



K FOLD Cross Validation

In [92]:

```
kfold_cv(x2,y_train)
```

```
Optimal alpha : 10
CV SCORE  0.849093751658438
Optimal alpha : 20
CV SCORE  0.849093751658438
Optimal alpha : 35
CV SCORE  0.849093751658438
Optimal alpha : 45
CV SCORE  0.849093751658438
Optimal alpha : 50
CV SCORE  0.849093751658438
Optimal alpha : 100
CV SCORE  0.849093751658438
```

GridsearchCV

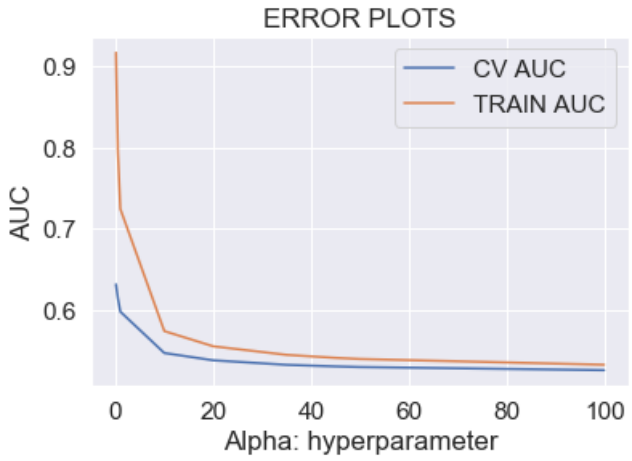
In [93]:

```
grid_search(x2,y_train,cv2,y_cv)
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[CV] alpha=0.1 .......................................................
[CV] ............... alpha=0.1, score=0.628092062897714, total=   0.0s

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s

[CV] alpha=0.1 .......................................................
[CV] ............... alpha=0.1, score=0.6314729967623958, total=   0.0s

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.2s remaining:    0.0s

[CV] alpha=0.1 .......................................................
[CV] ............... alpha=0.1, score=0.6344671030027147, total=   0.0s

[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.4s remaining:    0.0s

[CV] alpha=0.5 .......................................................
[CV] ............... alpha=0.5, score=0.6071358963966652, total=   0.0s

[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.6s remaining:    0.0s

[CV] alpha=0.5 .......................................................
[CV] ............... alpha=0.5, score=0.615620668821412, total=   0.0s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.7s remaining:    0.0s

[CV] alpha=0.5 .......................................................
[CV] ............... alpha=0.5, score=0.6195513491550956, total=   0.0s

[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.9s remaining:    0.0s

[CV] alpha=1 .........................................................
[CV] ................. alpha=1, score=0.5896046913371971, total=   0.0s

[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    1.1s remaining:    0.0s

[CV] alpha=1 .........................................................
[CV] ................. alpha=1, score=0.6003792406545982, total=   0.0s

[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    1.2s remaining:    0.0s

[CV] alpha=1 .........................................................
[CV] ................. alpha=1, score=0.6027854936826108, total=   0.0s

[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    1.4s remaining:    0.0s

[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.5402997715640832, total=   0.0s

[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    1.6s remaining:    0.0s

[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.5502363567660167, total=   0.0s

[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    1.8s remaining:    0.0s

[CV] alpha=10 ........................................................
[CV] ............... alpha=10, score=0.5492563395710874, total=   0.0s
```

```
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    2.0s remaining:    0.0s

[CV] alpha=20 ........................................................
[CV] ................ alpha=20, score=0.531986788983779, total=   0.0s

[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    2.2s remaining:    0.0s

[CV] alpha=20 ........................................................
[CV] ............... alpha=20, score=0.5411720250696589, total=   0.0s

[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    2.3s remaining:    0.0s

[CV] alpha=20 ........................................................
[CV] ............... alpha=20, score=0.5396490344659913, total=   0.0s
[CV] alpha=35 ........................................................
[CV] ............... alpha=35, score=0.5268891225139335, total=   0.0s
[CV] alpha=35 ........................................................
[CV] ............... alpha=35, score=0.5354749114819395, total=   0.0s
[CV] alpha=35 ........................................................
[CV] ............... alpha=35, score=0.5337539413560124, total=   0.0s
[CV] alpha=45 ........................................................
[CV] ............... alpha=45, score=0.5250426388565997, total=   0.0s
[CV] alpha=45 ........................................................
[CV] ................ alpha=45, score=0.533467130533536, total=   0.0s
[CV] alpha=45 ........................................................
[CV] ............... alpha=45, score=0.5316830846802514, total=   0.0s
[CV] alpha=50 ........................................................
[CV] ............... alpha=50, score=0.5243030852907432, total=   0.0s
[CV] alpha=50 ........................................................
[CV] ............... alpha=50, score=0.5326734683100885, total=   0.0s
[CV] alpha=50 ........................................................
[CV] .................. alpha=50, score=0.5308943502833, total=   0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.5204292757326443, total=   0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.5289200145112699, total=   0.0s
[CV] alpha=100 .......................................................
[CV] .............. alpha=100, score=0.5269679708918991, total=   0.0s

[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    4.5s finished

Optimal alpha  {'alpha': 0.1}
AUC:  0.631343956625667
```



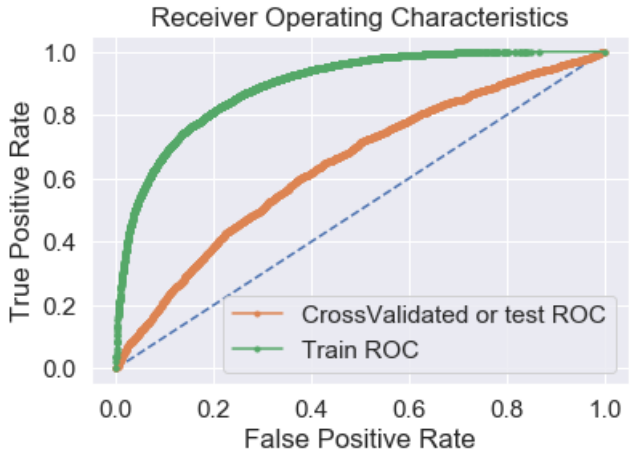

```
Accuracy on crossvalidated data:  0.6600023587808055
```

ROC Score on Test Data

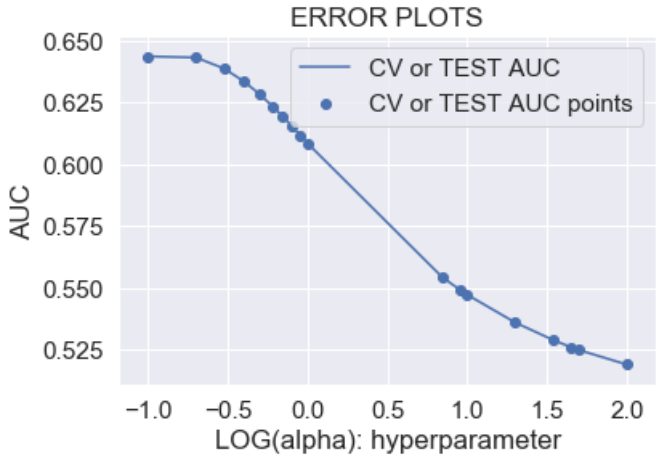
In [94]:

```
alpha from reg(x2, y_train_t2, y_test)
```

```
alpha_from_roc(x2,y_train,t2,y_test)
```




```
Optimal alpha  0.1
AUC:  0.6436257811354305
```



Confusion Matrix

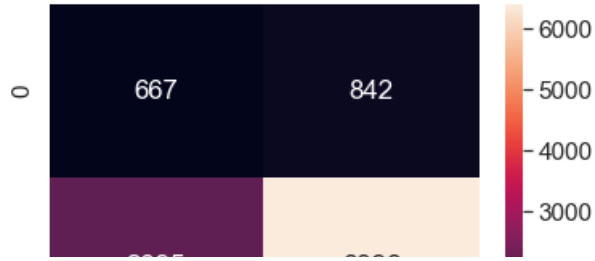
In [95]:

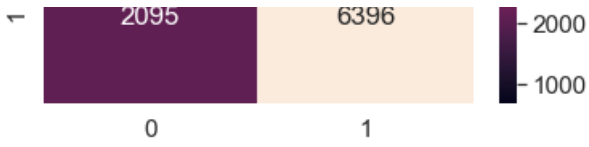
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import  MultinomialNB
labels=[0,1]
print("Test confusion matrix")
cm=confusion_matrix(y_test, predict(x2, y_train, t2, y_test,0.1),labels)
sns.set(font_scale=1.4)
sns.heatmap(cm,fmt='d',annot=True)
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.6497143609139692 for threshold 0.87
train AUC = 0.8951695137974744
```

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x658eca58>
```





2.4.2.1 Top 10 important features of positive class from SET 2

In [96]:

```
# Please write all the code with proper documentation
# alpha is selected based on k-fold CV score
from sklearn.naive_bayes import  MultinomialNB
clf = MultinomialNB(alpha=10)
clf.fit(x2,y_train)
important_feature_pos = pd.DataFrame(
    {'features': feature_bow,
     'prob_value': clf.feature_log_prob_[0],
    })

top_10_pos=important_feature_pos.sort_values(['prob_value'], ascending=[True]).tail(10)
print("Top 10  features for positive class")
print(top_10_pos)
```

```
Top 10  features for positive class
                features  prob_value
6           Health_Sports   -6.723719
29877            students   -6.709923
35           SpecialNeeds   -6.674542
5            SpecialNeeds   -6.674542
43                     CA   -6.654618
36     Literature_Writing   -6.337406
38               Literacy   -5.986827
37            Mathematics   -5.983719
7            Math_Science   -5.562603
8       Literacy_Language   -5.509071
```

2.4.2.2 Top 10 important features of negative class from SET 2

In [97]:

```
# Please write all the code with proper documentation
important_feature_neg = pd.DataFrame(
    {'features': feature_bow,
     'prob_value': clf.feature_log_prob_[1],
    })

top_10_neg=important_feature_neg.sort_values(['prob_value'], ascending=[True]).tail(10)
print("Top 10  features for negative class")
print(top_10_neg)
```

```
Top 10  features for negative class
                features  prob_value
5            SpecialNeeds   -5.495525
35           SpecialNeeds   -5.495525
6           Health_Sports   -5.471914
29877            students   -5.467054
43                     CA   -5.372834
36     Literature_Writing   -5.011992
37            Mathematics   -4.782901
38               Literacy   -4.566036
7            Math_Science   -4.406673
8       Literacy_Language   -4.138277
```

3. Conclusions

In [53]:

```
# Please compare all your models using Prettytable library

from prettytable import PrettyTable
x=PrettyTable()
x.field_names=["Vectorizer","Model","Hyperparameter","Test AUC"]
x.add_row(["BOW","MultinomialNB","0.5","0.694" ])
x.add_row(["TFIDF","MultinomialNB","0.1","0.643"])
print(x)
```

```
+------------+---------------+-----------------+-----------+
| Vectorizer |     Model     | Hyperparameter | Test AUC |
+------------+---------------+-----------------+-----------+
|    BOW     | MultinomialNB |      0.5        |  0.694    |
|   TFIDF    | MultinomialNB |      0.1        |  0.643    |
+------------+---------------+-----------------+-----------+
```

In [ ]:

```
```