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 [1]:

```
%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\Santosh\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
2. **The hyper paramter tuning(find best Alpha)**
   - Find the best hyper parameter which will give the maximum AUC value
   - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
   - Find the best hyper paramter using k-fold cross validation or simple cross validation data
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning
3. **Feature importance**
   - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB and print their corresponding feature names
4. **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. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
   - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
   - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.
5. **Conclusion**
   - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

1.1 Reading Data

In [2]:

```
project_data=pd.read_csv('train_data.csv', nrows=50000)
resource_data=pd.read_csv('resources.csv')
```

In [3]:

```
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 (50000, 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 [4]:

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

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

Out[4]:

	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

In [5]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data=resource_data.groupby('id').agg({'price':'sum','quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[5]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [6]:

```
# join two dataframes in python:
project_data=pd.merge(project_data, price_data, on='id', how='left')
```

In [7]:

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

Out[7]:

	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 [8]:

```
#presence of the numerical digits in a strings with numeric :
https://stackoverflow.com/a/19859308/8089731
def hasNumbers(inputString):
    return any(i.isdigit()for i in inputString)
p1=project_data[['id','project_resource_summary']]
p1=pd.DataFrame(data=p1)
p1.columns=['id','digits_in_summary']
p1['digits_in_summary']=p1['digits_in_summary'].map(hasNumbers)

# https://stackoverflow.com/a/17383325/8089731
p1['digits_in_summary'] = p1['digits_in_summary'].astype(int)
project_data=pd.merge(project_data,p1,on='id',how='left')
project_data.head(5)
```

Out[8]:

	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
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

1.2 preprocessing of project_subject_categories

In [9]:

```
categories=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 categories:
    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 & Hu
nger"]
```

```
        if 'The' in j.split():# this will split each of the catogory based on space "Math & 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:"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)
project_data.head(5)
```

Out[9]:

	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
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

In [10]:

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

Out[10]:

```
Counter({'Literacy_Language': 23998,
         'History_Civics': 2689,
         'Health_Sports': 6538,
         'Math_Science': 18874,
         'SpecialNeeds': 6233,
         'AppliedLearning': 5569,
         'Music_Arts': 4699,
         'Warmth': 643,
         'Care_Hunger': 643})
```

In [11]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
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 [12]:

```
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())
```

In [13]:

```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[13]:

	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 [14]:

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

In [15]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
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 [16]:

```
# 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 [17]:

```
# 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 [18]:

```
# 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', '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 [19]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    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 for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 50000/50000
[00:36<00:00, 1382.65it/s]
```

In [20]:

```
preprocessed_essays[2000]
```

Out[20]:

'describing students not easy task many would say inspirational creative hard working they unique unique interests learning abilities much what common desire learn day despite difficulties encounter our classroom amazing understand everyone learns pace as teacher i pride making sure stu dents always engaged motivated inspired create learning this project help students choose seating appropriate developmentally many students tire sitting chairs lessons different seats available he lps keep engaged learning flexible seating important classroom many students struggle attention fo cus engagement we currently stability balls seating well regular chairs stools help students trouble balance find difficult sit stability ball long period time we excited try stools part enga ging classroom community nannan'

In [21]:

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

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:01<00:00, 31079.03it/s]
```

In [22]:

```
preprocessed_titles[2000]
```

Out[22]:

'steady stools active learning'

In [23]:

```
project_grade_catogories = list(project_data['project_grade_category'].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

project_grade_cat_list = []
for i in tqdm(project_grade_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('&','_')
    project_grade_cat_list.append(temp.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:00<00:00, 382847.76it/s]
```

In [24]:

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

Out[24]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_title	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	I E
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Soccer Equipment for AWESOME Middle School Stu...	cl a
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Techie Kindergarteners	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Interactive Math Tools	g r

5 rows × 21 columns

In [25]:

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

Out[25]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_title	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	M
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Soccer Equipment for AWESOME Middle School Stu...	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Techie Kindergarteners	M

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_title	pr
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Interactive Math Tools	

In [26]:

```
project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_titles'] = preprocessed_titles
```

In [27]:

```
#Replacing Nan's with maximum occured value: https://stackoverflow.com/a/51053916/8089731
project_data['teacher_prefix'].value_counts().argmax()
project_data.fillna(value=project_data['teacher_prefix'].value_counts().argmax(),axis=1,inplace=True)
```

1.5 Preparing data for models

In [28]:

```
project_data.columns
```

Out[28]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'digits_in_summary', 'clean_categories',
       'clean_subcategories', 'essay', 'clean_project_grade_category',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

In [29]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection
```

2. Naive Bayes

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

In [30]:

```
# 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
```

In [31]:

```
X_train, X_test, y_train, y_test = train_test_split(project_data,project_data['project_is_approved'
], test_size=0.33, stratify = project_data['project_is_approved'])
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)

X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [32]:

```
# 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
```

Vectorizing Categorical data

one hot encodig

In [33]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(lowercase=False, binary=True)
vectorizer_cat.fit(X_train['clean_categories'].values)
print(vectorizer_cat.get_feature_names())


categories_one_hot_xtrain = vectorizer_cat.transform(X_train['clean_categories'].values)
categories_one_hot_xcv = vectorizer_cat.transform(X_cv['clean_categories'].values)
categories_one_hot_xtest = vectorizer_cat.transform(X_test['clean_categories'].values)
print("Shape of matrix after one hot encodig_xtrain ",categories_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",categories_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",categories_one_hot_xtest.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix after one hot encodig_xtrain  (22445, 9)
Shape of matrix after one hot encodig_xcv  (11055, 9)
Shape of matrix after one hot encodig_xtest  (16500, 9)
```

In [34]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_sub_cat = CountVectorizer(lowercase=False, binary=True)
vectorizer_sub_cat.fit(X_train['clean_subcategories'].values)
print(vectorizer_sub_cat.get_feature_names())


sub_categories_one_hot_xtrain = vectorizer_sub_cat.transform(X_train['clean_subcategories'].values
)
sub_categories_one_hot_xcv = vectorizer_sub_cat.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_xtest = vectorizer_sub_cat.transform(X_test['clean_subcategories'].values)
print("Shape of matrix after one hot encodig_xtrain ",sub_categories_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",sub_categories_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",sub_categories_one_hot_xtest.shape)
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
```

```
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
Shape of matrix after one hot encodig_xtrain  (22445, 30)
Shape of matrix after one hot encodig_xcv  (11055, 30)
Shape of matrix after one hot encodig_xtest  (16500, 30)
```

In [35]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_state = CountVectorizer( lowercase=False, binary=True)
vectorizer_state.fit(X_train['school_state'].values)
print(vectorizer_state.get_feature_names())


school_state_one_hot_xtrain = vectorizer_state.transform(X_train['school_state'].values)
school_state_one_hot_xcv = vectorizer_state.transform(X_cv['school_state'].values)
school_state_one_hot_xtest = vectorizer_state.transform(X_test['school_state'].values)
print("Shape of matrix after one hot encodig_train ",school_state_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_cv ",school_state_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_test ",school_state_one_hot_xtest.shape)
```

```
['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_train  (22445, 51)
Shape of matrix after one hot encodig_cv  (11055, 51)
Shape of matrix after one hot encodig_test  (16500, 51)
```

In [36]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_teacherprefix = CountVectorizer( lowercase=False, binary=True)
vectorizer_teacherprefix.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer_teacherprefix.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
teacher_prefix_one_hot_xtrain =
vectorizer_teacherprefix.transform(X_train['teacher_prefix'].values.astype('U'))
teacher_prefix_one_hot_xcv =
vectorizer_teacherprefix.transform(X_cv['teacher_prefix'].values.astype('U'))
teacher_prefix_one_hot_xtest = vectorizer_teacherprefix.transform(X_test['teacher_prefix'].values.a
stype('U'))
print("Shape of matrix after one hot encodig_xtrain ",teacher_prefix_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",teacher_prefix_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",teacher_prefix_one_hot_xtest.shape)
```

```
['Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig_xtrain  (22445, 4)
Shape of matrix after one hot encodig_xcv  (11055, 4)
Shape of matrix after one hot encodig_xtest  (16500, 4)
```

In [37]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
# https://stackoverflow.com/a/38161028/8089731
pattern = "(?u)\\b[\\w-]+\\b"
vectorizer_projectgrade = CountVectorizer(token_pattern=pattern, lowercase=False, binary=True)
vectorizer_projectgrade.fit(X_train['clean_project_grade_category'].values)
print(vectorizer_projectgrade.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
project_grade_cat_one_hot_xtrain =
vectorizer_projectgrade.transform(X_train['clean_project_grade_category'].values)
project_grade_cat_one_hot_xcv =
vectorizer_projectgrade.transform(X_cv['clean_project_grade_category'].values)
project_grade_cat_one_hot_xtest =
vectorizer_projectgrade.transform(X_test['clean_project_grade_category'].values)
```

```
print("Shape of matrix after one hot encodig_xtrain ",project_grade_cat_one_hot_xtrain.shape)
print("Shape of matrix after one hot encodig_xcv ",project_grade_cat_one_hot_xcv.shape)
print("Shape of matrix after one hot encodig_xtest ",project_grade_cat_one_hot_xtest.shape)
```

```
['Grades3-5', 'Grades6-8', 'Grades9-12', 'GradesPreK-2']
Shape of matrix after one hot encodig_xtrain  (22445, 4)
Shape of matrix after one hot encodig_xcv  (11055, 4)
Shape of matrix after one hot encodig_xtest  (16500, 4)
```

Vectorizing Numerical features

In [50]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
#from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import Normalizer
# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(1, -1)

price_scalar = Normalizer()
price_scalar.fit(X_train['price'].values.reshape(1, -1)) # finding the mean and standard deviation
of this data
# print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized_xtrain = price_scalar.transform(X_train['price'].values.reshape(1, -1))
price_standardized_xcv = price_scalar.transform(X_cv['price'].values.reshape(1, -1))
price_standardized_xtest = price_scalar.transform(X_test['price'].values.reshape(1, -1))

# reshaping again with reshape(-1, 1)
price_standardized_xtrain=price_standardized_xtrain.reshape(-1, 1)
price_standardized_xcv=price_standardized_xcv.reshape(-1, 1)
price_standardized_xtest=price_standardized_xtest.reshape(-1, 1)


print("shape of price_standardized_xtrain",price_standardized_xtrain.shape)
print("shape of price_standardized_xcv",price_standardized_xcv.shape)
print("shape of price_standardized_xtest",price_standardized_xtest.shape)
```

```
shape of price_standardized_xtrain (22445, 1)
shape of price_standardized_xcv (11055, 1)
shape of price_standardized_xtest (16500, 1)
```

In [51]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import Normalizer

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(1,-1)

quantity_scalar = Normalizer()
quantity_scalar.fit(X_train['quantity'].values.reshape(1,-1)) # finding the mean and standard
deviation of this data
# print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation :
{np.sqrt(quantity_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
quantity_standardized_xtrain = quantity_scalar.transform(X_train['quantity'].values.reshape(1,-1))
quantity_standardized_xcv = quantity_scalar.transform(X_cv['quantity'].values.reshape(1,-1))
quantity_standardized_xtest = quantity_scalar.transform(X_test['quantity'].values.reshape(1,-1))
```

```
quantity_standardized_xtest = quantity_scalar.transform(X_test['quantity'].values.reshape(1, 1))


# reshaping again with reshape(-1,1)
quantity_standardized_xtrain=quantity_standardized_xtrain.reshape(-1,1)
quantity_standardized_xcv=quantity_standardized_xcv.reshape(-1,1)
quantity_standardized_xtest=quantity_standardized_xtest.reshape(-1,1)


print("shape of quantity_standardized_xtrain",quantity_standardized_xtrain.shape)
print("shape of quantity_standardized_xcv",quantity_standardized_xcv.shape)
print("shape of quantity_standardized_xtest",quantity_standardized_xtest.shape)
```

```
shape of quantity_standardized_xtrain (22445, 1)
shape of quantity_standardized_xcv (11055, 1)
shape of quantity_standardized_xtest (16500, 1)
```

In [53]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import Normalizer

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(1, -1)

teacher_num_prev_projects_scalar = Normalizer()
teacher_num_prev_projects_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].value
s.reshape(-1, 1)) # finding the mean and standard deviation of this data
# print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_[0]}, Standard deviatio
n : {np.sqrt(teacher_number_of_previously_posted_projects_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
teacher_num_prev_projects_standardized_xtrain = teacher_num_prev_projects_scalar.transform(X_train
['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
teacher_num_prev_projects_standardized_xcv = teacher_num_prev_projects_scalar.transform(X_cv['teac
her_number_of_previously_posted_projects'].values.reshape(1,-1))
teacher_num_prev_projects_standardized_xtest = teacher_num_prev_projects_scalar.transform(X_test['
teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

# reshaping again with reshape(-1,1)
teacher_num_prev_projects_standardized_xtrain=teacher_num_prev_projects_standardized_xtrain.reshap
e(-1, 1)
teacher_num_prev_projects_standardized_xcv=teacher_num_prev_projects_standardized_xcv.reshape(-1,
1)
teacher_num_prev_projects_standardized_xtest=teacher_num_prev_projects_standardized_xtest.reshape(
-1, 1)


print(" shape of
teacher_number_of_previously_posted_projects_standardized_xtrain",teacher_num_prev_projects_standar
ized_xtrain.shape)
print(" shape of
teacher_number_of_previously_posted_projects_standardized_xcv",teacher_num_prev_projects_standardiz
d_xcv.shape)
print(" shape of
teacher_number_of_previously_posted_projects_standardized_xtest",teacher_num_prev_projects_standarc
zed_xtest.shape)
```

```
 shape of teacher_number_of_previously_posted_projects_standardized_xtrain (22445, 1)
 shape of teacher_number_of_previously_posted_projects_standardized_xcv (11055, 1)
 shape of teacher_number_of_previously_posted_projects_standardized_xtest (16500, 1)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [54]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
```

```
# 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
```

Vectorizing Text data

BOW on eassay

In [55]:

```
# BOW on eassay
# We are considering only the words which appeared in at least 10 documents(rows or projects).

vectorizer_bow_essays = CountVectorizer(min_df=10)
vectorizer_bow_essays.fit(X_train['preprocessed_essays'])

essay_text_bow_xtrain = vectorizer_bow_essays.transform(X_train['preprocessed_essays'])
essay_text_bow_xcv = vectorizer_bow_essays.transform(X_cv['preprocessed_essays'])
essay_text_bow_xtest = vectorizer_bow_essays.transform(X_test['preprocessed_essays'])

print("Shape of matrix after BOW_text_essay X_train ",essay_text_bow_xtrain.shape)
print("Shape of matrix after BOW_text_essay X_cv ",essay_text_bow_xcv.shape)
print("Shape of matrix after BOW_text_essay X_test ",essay_text_bow_xtest.shape)
```

```
Shape of matrix after BOW_text_essay X_train  (22445, 8859)
Shape of matrix after BOW_text_essay X_cv  (11055, 8859)
Shape of matrix after BOW_text_essay X_test  (16500, 8859)
```

BOW on project_title

In [56]:

```
# BOW on project_title
# We are considering only the words which appeared in at least 10 documents(rows or projects).

vectorizer_bow_titles = CountVectorizer(min_df=10)
vectorizer_bow_titles.fit(X_train['preprocessed_titles'])

proj_title_bow_xtrain = vectorizer_bow_titles.transform(X_train['preprocessed_titles'])
proj_title_bow_xcv = vectorizer_bow_titles.transform(X_cv['preprocessed_titles'])
proj_title_bow_xtest = vectorizer_bow_titles.transform(X_test['preprocessed_titles'])

print("Shape of matrix after BOW project_title_xtrain ",proj_title_bow_xtrain.shape)
print("Shape of matrix after BOW project_title_xcv ",proj_title_bow_xcv.shape)
print("Shape of matrix after BOW project_title_xtest ",proj_title_bow_xtest.shape)
```

```
Shape of matrix after BOW project_title_xtrain  (22445, 1231)
Shape of matrix after BOW project_title_xcv  (11055, 1231)
Shape of matrix after BOW project_title_xtest  (16500, 1231)
```

TFIDF Vectorizer on Essay

In [57]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essays = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essays.fit(X_train['preprocessed_essays'])

essay_tfidf_xtrain = vectorizer_tfidf_essays.transform(X_train['preprocessed_essays'])
essay_tfidf_xcv = vectorizer_tfidf_essays.transform(X_cv['preprocessed_essays'])
```

```
essay_tfidf_xtest = vectorizer_tfidf_essays.transform(X_test['preprocessed_essays'])

print("Shape of matrix after tfidf eassay_xtrain ",essay_tfidf_xtrain.shape)
print("Shape of matrix after tfidf essay_xcv ",essay_tfidf_xcv.shape)
print("Shape of matrix after tfidf essay_xtest ",essay_tfidf_xtest.shape)
```

```
Shape of matrix after tfidf eassay_xtrain  (22445, 8859)
Shape of matrix after tfidf essay_xcv  (11055, 8859)
Shape of matrix after tfidf essay_xtest  (16500, 8859)
```

TFIDF Vectorizer on Project Title

In [58]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(X_train['preprocessed_titles'])

proj_title_tfidf_xtrain = vectorizer_tfidf_title.transform(X_train['preprocessed_titles'])
proj_title_tfidf_xcv = vectorizer_tfidf_title.transform(X_cv['preprocessed_titles'])
proj_title_tfidf_xtest = vectorizer_tfidf_title.transform(X_test['preprocessed_titles'])

print("Shape of matrix after tfidf proj_title_xtrain ",proj_title_tfidf_xtrain.shape)
print("Shape of matrix after tfidf proj_title_xcv ",proj_title_tfidf_xcv.shape)
print("Shape of matrix after tfidf proj_title_xtest ",proj_title_tfidf_xtest.shape)
```

```
Shape of matrix after tfidf proj_title_xtrain  (22445, 1231)
Shape of matrix after tfidf proj_title_xcv  (11055, 1231)
Shape of matrix after tfidf proj_title_xtest  (16500, 1231)
```

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

2.4.1 Applying Naive Bayes on BOW, SET 1

In [59]:

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

In [60]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack


X_train1=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
school_state_one_hot_xtrain,
                 teacher_prefix_one_hot_xtrain, project_grade_cat_one_hot_xtrain,
price_standardized_xtrain,
               teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                essay_text_bow_xtrain, proj_title_bow_xtrain)).tocsr().toarray()


X_cv1=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
                school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
                project_grade_cat_one_hot_xcv, price_standardized_xcv,
               teacher_num_prev_projects_standardized_xcv, quantity_standardized_xcv,
                essay_text_bow_xcv, proj_title_bow_xcv)).tocsr().toarray()


X test1=hstack((categories one hot xtest, sub categories one hot xtest,
```

```
                school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
                project_grade_cat_one_hot_xtest, price_standardized_xtest,
               teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                essay_text_bow_xtest, proj_title_bow_xtest)).tocsr().toarray()

print(X_train1.shape, y_train.shape)
print(X_cv1.shape, y_cv.shape)
print(X_test1.shape, y_test.shape)
```

```
(22445, 10191) (22445,)
(11055, 10191) (11055,)
(16500, 10191) (16500,)
```

In [61]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train1 = scaler.fit_transform(X_train1,y_train)
X_cv1 = scaler.transform(X_cv1)
X_test1 = scaler.transform(X_test1)

print(X_train1.shape, y_train.shape)
print(X_cv1.shape, y_cv.shape)
print(X_test1.shape, y_test.shape)
```

```
(22445, 10191) (22445,)
(11055, 10191) (11055,)
(16500, 10191) (16500,)
```

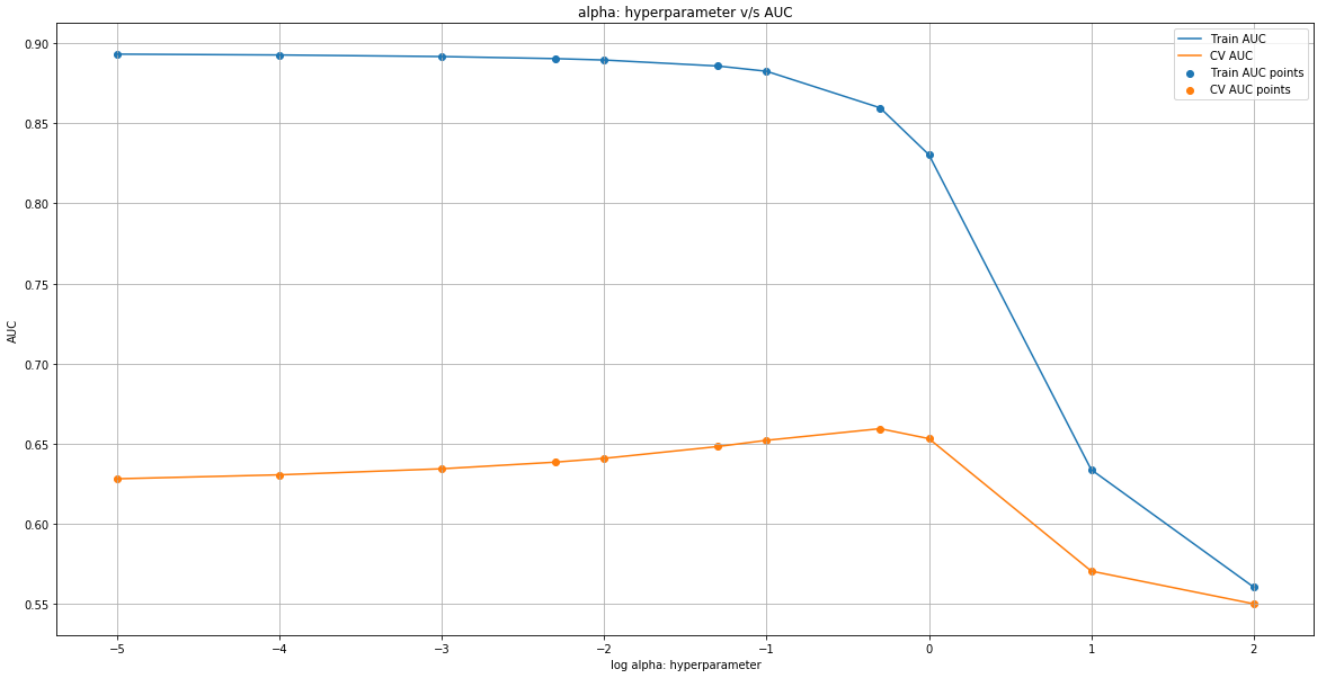
Random alpha values (hyperparameter)

In [62]:

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

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

alphas = [0.00001, 0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 10, 100]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior=[0.5,0.5])
    nb.fit(X_train1, y_train)

    y_train_pred = nb.predict_proba(X_train1)[:,1]
    y_cv_pred = nb.predict_proba(X_cv1)[:,1]


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs

    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for a in tqdm(alphas):
    b = np.log10(a)
    log_alphas.append(b)

log_alphas = np.array(log_alphas)
alphas = np.array(alphas)

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

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

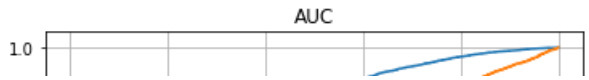
```
plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')

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

plt.grid()
plt.show()
```

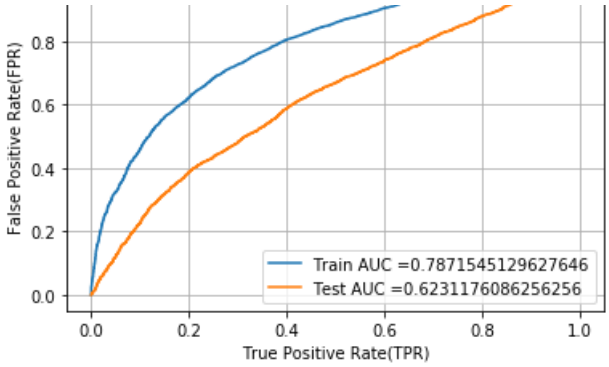
```
100%|██████████| 11/11 [00:37<00:00,  2.46s/it]
100%|██████████| 11/11 [00:00<00:00, 67.11it/s]
```



In [68]:

```
#
https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc
ve
from sklearn.metrics import roc_curve, auc

nb_bow = MultinomialNB(alpha = 1.8,class_prior=[0.5,0.5])
nb_bow.fit(X_train1, 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 = nb_bow.predict_proba(X_train1)[:,1]
y_test_pred = nb_bow.predict_proba(X_test1)[:,1]

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()
plt.show()
```





In [69]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

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

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

In [70]:

```
from sklearn.metrics import confusion_matrix

print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.416
[[ 1732  1731]
 [ 2706 16276]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.57
[[  996  1550]
 [ 3567 10387]]
```

Confusion matrix for train data

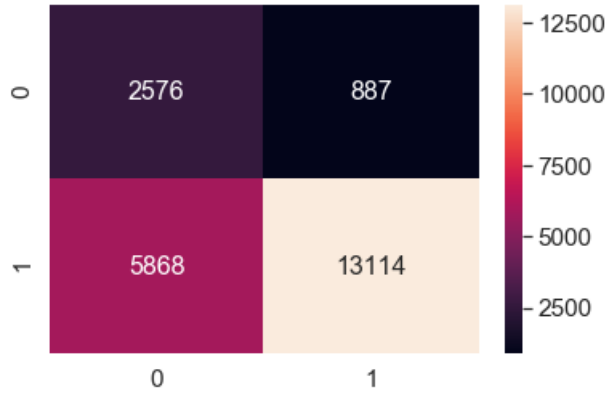
In [71]:

```
# Confusion matrix for train data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtrain =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred,
tr_thresholds, train_fpr, train_tpr)))

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

```
the maximum value of tpr*(1-fpr) 0.5139094188141156 for threshold 0.682
```

Out[71]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15f33037ac8>
```



Conclusion

1. True Positive Rate is High as well as False Positive Rate is also high whih is not desirable
2. so Using Bag of Words We have both TPR and FPR high

Confusion matrix for test data

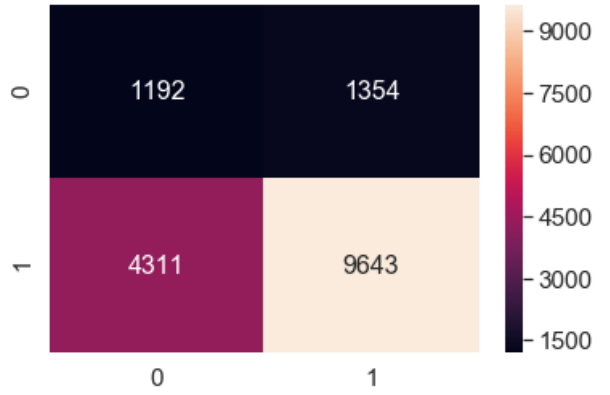
In [72]:

```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))

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

```
the maximum value of tpr*(1-fpr) 0.35329194645947565 for threshold 0.644
```

Out[72]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15f334a1ac8>
```



Conclusion

1.For Test Data using Bag of words vectorization Both TPR and FPR is HIgh
2.The result is not Desirable

2.4.1.1 Top 10 important features of positive class from SET 1

In [73]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack


X_train1=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
school_state_one_hot_xtrain,
                 teacher_prefix_one_hot_xtrain, project_grade_cat_one_hot_xtrain,
price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                 essay_text_bow_xtrain, proj_title_bow_xtrain)).tocsr().toarray()


X_cv1=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
                 school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
                 project_grade_cat_one_hot_xcv, price_standardized_xcv,
                teacher_num_prev_projects_standardized_xcv, quantity_standardized_xcv,
                 essay_text_bow_xcv, proj_title_bow_xcv)).tocsr().toarray()


X_test1=hstack((categories_one_hot_xtest, sub_categories_one_hot_xtest,
                 school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
                 project_grade_cat_one_hot_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                 essay_text_bow_xtest, proj_title_bow_xtest)).tocsr().toarray()

print(X_train1.shape, y_train.shape)
print(X_cv1.shape, y_cv.shape)
print(X_test1.shape, y_test.shape)
```

```
(22445, 10191) (22445,)
(11055, 10191) (11055,)
(16500, 10191) (16500,)
```

In [74]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train1 = scaler.fit_transform(X_train1,y_train)
X_cv1 = scaler.transform(X_cv1)
X_test1 = scaler.transform(X_test1)

print(X_train1.shape, y_train.shape)
print(X_cv1.shape, y_cv.shape)
print(X_test1.shape, y_test.shape)
```

```
(22445, 10191) (22445,)
(11055, 10191) (11055,)
(16500, 10191) (16500,)
```

In [75]:

```
from sklearn.naive_bayes import MultinomialNB


NB_bow_FI = MultinomialNB(alpha = 1,class_prior=[0.5,0.5])
NB_bow_FI.fit(X_train1, y_train)
```

Out[75]:

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

In [76]:

```
bow_features_probs_pos = []
```

```
for a in range(10191) :
    b = NB_bow_FI.feature_log_prob_[1,a]
    bow_features_probs_pos.append(b)

len(bow_features_probs_pos)
```

Out[76]:

```
10191
```

In [77]:

```
bow_imp_features = []

for a in vectorizer_cat.get_feature_names() :
    bow_imp_features.append(a)

for a in vectorizer_sub_cat.get_feature_names() :
    bow_imp_features.append(a)

for a in vectorizer_state.get_feature_names() :
    bow_imp_features.append(a)

for a in vectorizer_teacherprefix.get_feature_names() :
    bow_imp_features.append(a)

for a in vectorizer_projectgrade.get_feature_names() :
    bow_imp_features.append(a)

bow_imp_features.append("price")
bow_imp_features.append("quantity")
bow_imp_features.append("teacher_number_of_previously_posted")

for a in vectorizer_bow_essays.get_feature_names() :
    bow_imp_features.append(a)

for a in vectorizer_bow_titles.get_feature_names() :
    bow_imp_features.append(a)
```

In [78]:

```
len(bow_imp_features)
```

Out[78]:

```
10191
```

In [79]:

```
final_features_bow_imp_pos = pd.DataFrame({'feature_prob_estimates' :
bow_features_probs_pos,'feature_names' : bow_imp_features})
final_features_bow_imp_pos.sort_values(by = ['feature_prob_estimates'], ascending = False,inplace=T
rue)
```

In [80]:

```
print("Top 10 Important features of positive class from SET1")

final_features_bow_imp_pos.head(10)
```

```
Top 10 Important features of positive class from SET1
```

Out[80]:

	feature_prob_estimates	feature_names
5360	-3.666130	nannan
91	-4.264629	Mrs
4	-4.330037	Literacy_Language
97	-4.530672	GradesPreK-2

	feature_prob_estimates	feature_names
5	-4.611206	Math_Science
92	-4.642776	Ms
94	-4.684801	Grades3-5
26	-4.756832	Literacy
7771	-4.845946	students
28	-4.986480	Mathematics

2.4.1.2 Top 10 important features of negative class from SET 1

In [81]:

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

bow_features_probs_neg = []
for c in range(10191) :
    d = nb_bow.feature_log_prob_[0,c]
    bow_features_probs_neg.append(d)
```

In [82]:

```
final_features_bow_imp_neg = pd.DataFrame({'feature_prob_estimates' :
bow_features_probs_neg,'feature_names' : bow_imp_features})
final_features_bow_imp_neg.sort_values(by = ['feature_prob_estimates'], ascending = False,inplace=T
rue)
```

In [83]:

```
print("Top 10 Important features of negative class from SET1")

final_features_bow_imp_neg.head(10)
```

```
Top 10 Important features of negative class from SET1
```

Out[83]:

	feature_prob_estimates	feature_names
5360	-3.755774	nannan
91	-4.389726	Mrs
4	-4.539517	Literacy_Language
97	-4.612612	GradesPreK-2
5	-4.632117	Math_Science
92	-4.721333	Ms
94	-4.849660	Grades3-5
7771	-4.997627	students
26	-5.037655	Literacy
28	-5.063147	Mathematics

2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [84]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_train2=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
                 school_state_one_hot_xtrain, teacher_prefix_one_hot_xtrain,
                 project_grade_cat_one_hot_xtrain, price_standardized_xtrain,
```

```
                 teacher_num_prev_projects_standardized_xtrain,
                 quantity_standardized_xtrain,essay_tfidf_xtrain, proj_title_tfidf_xtrain)).tocsr()
.toarray()


X_cv2=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
               school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
               project_grade_cat_one_hot_xcv, price_standardized_xcv,
              teacher_num_prev_projects_standardized_xcv,quantity_standardized_xcv,
               essay_tfidf_xcv, proj_title_tfidf_xcv)).tocsr().toarray()


X_test2=hstack((categories_one_hot_xtest, sub_categories_one_hot_xtest,
               school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
               project_grade_cat_one_hot_xtest, price_standardized_xtest,
              teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
               essay_tfidf_xtest, proj_title_tfidf_xtest)).tocsr().toarray()

print(X_train2.shape)
print(X_cv2.shape)
print(X_test2.shape)
```

```
(22445, 10191)
(11055, 10191)
(16500, 10191)
```

In [85]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train2 = scaler.fit_transform(X_train2,y_train)
X_cv2 = scaler.transform(X_cv2)
X_test2 = scaler.transform(X_test2)

print(X_train2.shape, y_train.shape)
print(X_cv2.shape, y_cv.shape)
print(X_test2.shape, y_test.shape)
```

```
(22445, 10191) (22445,)
(11055, 10191) (11055,)
(16500, 10191) (16500,)
```

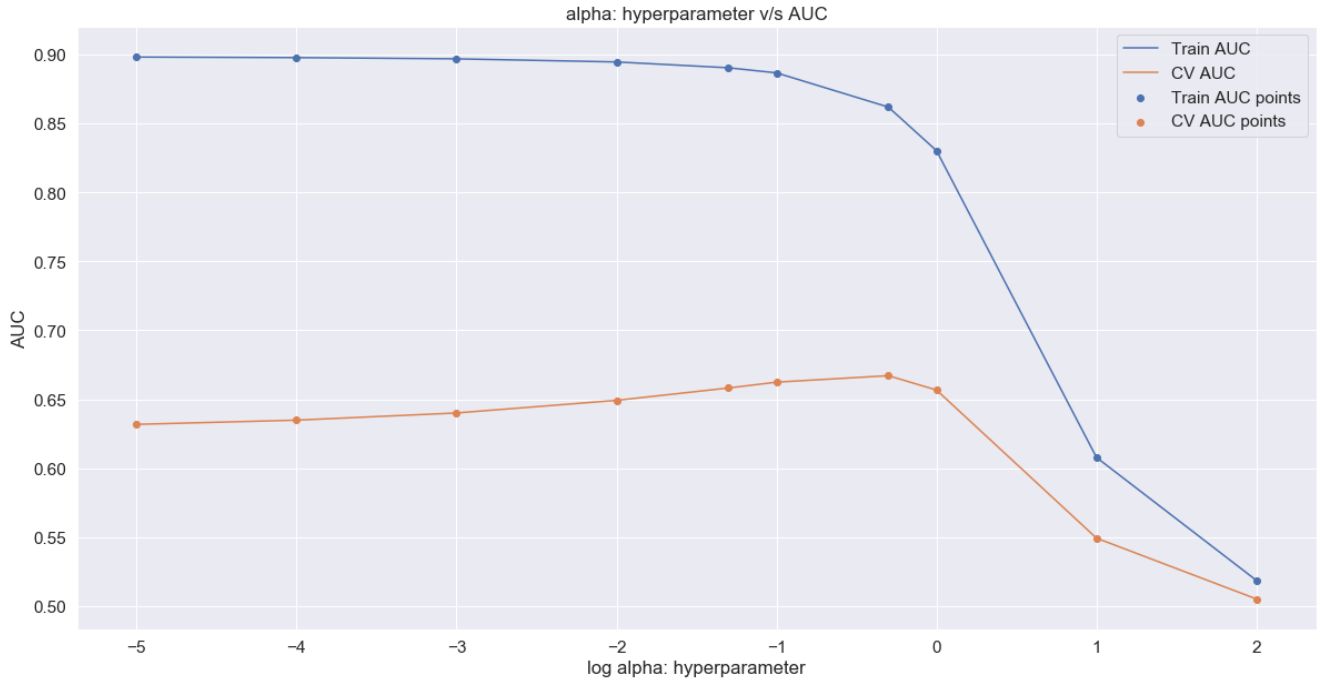
Random alpha values (hyperparameter)

In [86]:

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

train_auc = []
cv_auc = []
log_alphas = []
alphas = [0.00001, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1, 10, 100]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior=[0.5,0.5])
    nb.fit(X_train2, y_train)

    y_train_pred = nb.predict_proba(X_train2)[:,1]
    y_cv_pred = nb.predict_proba(X_cv2)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs

    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

```
for a in tqdm(alphas):
    b = np.log10(a)
    log_alphas.append(b)

log_alphas = np.array(log_alphas)
alphas = np.array(alphas)

plt.figure(figsize=(20,10))
plt.grid()
plt.plot(log_alphas, train_auc, label='Train AUC')
plt.plot(log_alphas, cv_auc, label='CV AUC')

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

plt.legend()

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

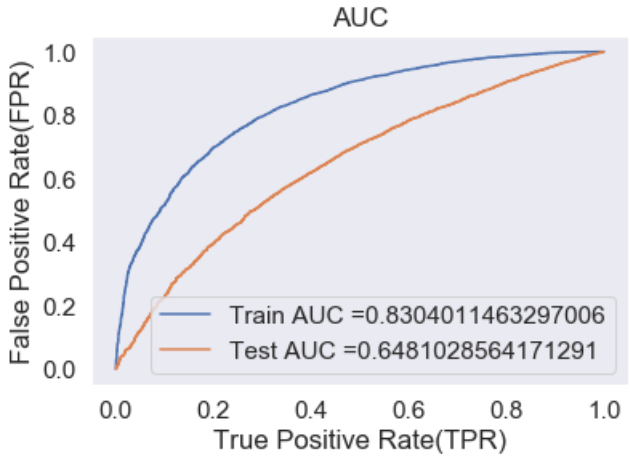
```
100%|██████████| 10/10 [00:25<00:00,  2.56s/it]
100%|██████████| 10/10 [00:00<00:00, 60.94it/s]
```



In [89]:

```
#
https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc
ve
from sklearn.metrics import roc_curve, auc

nb_tfidf = MultinomialNB(alpha = 1,class_prior=[0.5,0.5])
nb_tfidf.fit(X_train2, 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 = nb_tfidf.predict_proba(X_train2)[:,1]
y_test_pred = nb_tfidf.predict_proba(X_test2)[:,1]

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()
plt.show()
```



In [90]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

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

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

In [91]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5644457978990872 for threshold 0.526
[[ 2584   879]
 [ 4623 14359]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3742950268309486 for threshold 0.524
[[ 1235  1311]
 [ 3919 10035]]
```

In [92]:

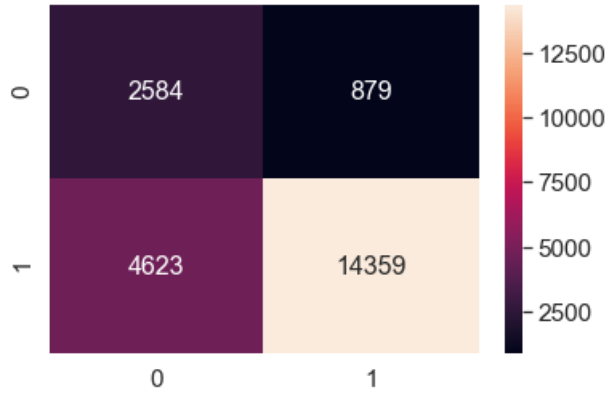
```
# Confusion Matrix for Train Data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtrain =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred,
tr_thresholds, train_fpr, train_tpr)))
```

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

```
the maximum value of tpr*(1-fpr) 0.5644457978990872 for threshold 0.526
```

Out[92]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15f33a0d5c0>
```



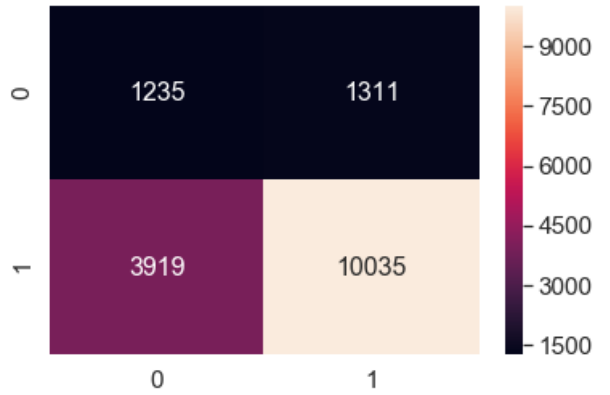
In [93]:

```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))

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

```
the maximum value of tpr*(1-fpr) 0.3742950268309486 for threshold 0.524
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15f3442e9e8>
```



2.4.2.1 Top 10 important features of positive class from SET 2

In [94]:

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

# Please write all the code with proper documentation

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
```

```
X_train2=hstack((categories_one_hot_xtrain, sub_categories_one_hot_xtrain,
                 school_state_one_hot_xtrain, teacher_prefix_one_hot_xtrain,
                 project_grade_cat_one_hot_xtrain, price_standardized_xtrain,
                  teacher_num_prev_projects_standardized_xtrain,
                  quantity_standardized_xtrain,essay_tfidf_xtrain, proj_title_tfidf_xtrain)).tocsr()
.toarray()


X_cv2=hstack((categories_one_hot_xcv, sub_categories_one_hot_xcv,
                 school_state_one_hot_xcv, teacher_prefix_one_hot_xcv,
                 project_grade_cat_one_hot_xcv, price_standardized_xcv,
                teacher_num_prev_projects_standardized_xcv,quantity_standardized_xcv,
                 essay_tfidf_xcv, proj_title_tfidf_xcv)).tocsr().toarray()


X_test2=hstack((categories_one_hot_xtest, sub_categories_one_hot_xtest,
                 school_state_one_hot_xtest, teacher_prefix_one_hot_xtest,
                 project_grade_cat_one_hot_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                 essay_tfidf_xtest, proj_title_tfidf_xtest)).tocsr().toarray()

print(X_train2.shape)
print(X_cv2.shape)
print(X_test2.shape)
```

```
(22445, 10191)
(11055, 10191)
(16500, 10191)
```

In [95]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train2 = scaler.fit_transform(X_train2,y_train)
X_cv2 = scaler.transform(X_cv2)
X_test2 = scaler.transform(X_test2)

print(X_train2.shape, y_train.shape)
print(X_cv2.shape, y_cv.shape)
print(X_test2.shape, y_test.shape)
```

```
(22445, 10191) (22445,)
(11055, 10191) (11055,)
(16500, 10191) (16500,)
```

In [96]:

```
from sklearn.naive_bayes import MultinomialNB


NB_tfidf_FI = MultinomialNB(alpha = 0.1,class_prior=[0.5,0.5])
NB_tfidf_FI.fit(X_train2, y_train)
```

Out[96]:

```
MultinomialNB(alpha=0.1, class_prior=[0.5, 0.5], fit_prior=True)
```

In [97]:

```
tfidf_features_probs_pos = []

for a in range(10191) :
    b = NB_tfidf_FI.feature_log_prob_[1,a]
    tfidf_features_probs_pos.append(b)

len(tfidf_features_probs_pos)
```

Out[97]:

```
10191
```

In [98]:

```
tfidf_imp_features = []

for a in vectorizer_cat.get_feature_names() :
    tfidf_imp_features.append(a)

for a in vectorizer_sub_cat.get_feature_names() :
    tfidf_imp_features.append(a)

for a in vectorizer_state.get_feature_names() :
    tfidf_imp_features.append(a)

for a in vectorizer_teacherprefix.get_feature_names() :
    tfidf_imp_features.append(a)

for a in vectorizer_projectgrade.get_feature_names() :
    tfidf_imp_features.append(a)

tfidf_imp_features.append("price")
tfidf_imp_features.append("quantity")
tfidf_imp_features.append("teacher_number_of_previously_posted")

for a in vectorizer_tfidf_essays.get_feature_names() :
    tfidf_imp_features.append(a)

for a in vectorizer_tfidf_title.get_feature_names() :
    tfidf_imp_features.append(a)
```

In [99]:

```
len(tfidf_imp_features)
```

Out[99]:

```
10191
```

In [100]:

```
final_features_tfidf_imp_pos = pd.DataFrame({'feature_prob_estimates' : tfidf_features_probs_pos,'
feature_names' : tfidf_imp_features})
final_features_tfidf_imp_pos.sort_values(by = ['feature_prob_estimates'], ascending =
False,inplace=True)
```

In [101]:

```
print("Top 10 Important features of positive class from SET2")

final_features_tfidf_imp_pos.head(10)
```

```
Top 10 Important features of positive class from SET2
```

Out[101]:

	feature_prob_estimates	feature_names
5360	-4.157201	nannan
91	-4.239068	Mrs
4	-4.304483	Literacy_Language
97	-4.505139	GradesPreK-2
7771	-4.583850	students
5	-4.585683	Math_Science
92	-4.617257	Ms
94	-4.659288	Grades3-5
26	-4.731329	Literacy
28	-4.961015	Mathematics

2.4.2.2 Top 10 important features of negative class from SET 2

In [102]:

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

tfidf_features_probs_neg = []
for c in range(10191) :
    d = nb_tfidf.feature_log_prob_[0,c]
    tfidf_features_probs_neg.append(d)
```

In [103]:

```
final_features_tfidf_imp_neg = pd.DataFrame({'feature_prob_estimates' : tfidf_features_probs_neg,'
feature_names' : tfidf_imp_features})
final_features_tfidf_imp_neg.sort_values(by = ['feature_prob_estimates'], ascending =
False,inplace=True)
```

In [104]:

```
print("Top 10 Important features of negative class from SET2")

final_features_tfidf_imp_neg.head(10)
```

```
Top 10 Important features of negative class from SET2
```

Out[104]:

	feature_prob_estimates	feature_names
5360	-4.188710	nannan
91	-4.351424	Mrs
4	-4.501289	Literacy_Language
97	-4.574425	GradesPreK-2
5	-4.593941	Math_Science
7771	-4.669708	students
92	-4.683211	Ms
94	-4.811625	Grades3-5
26	-4.999771	Literacy
28	-5.025286	Mathematics

3. Conclusions

In [105]:

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

In [106]:

```
from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter:Alpha", "AUC"]
x.add_row(["BOW", "Naive Bayes", 1.8, 0.62])
x.add_row(["TFIDF", "Naive Bayes", 1, 0.64])
print(x)
```

```
+------------+-------------+-----------------------+------+
| Vectorizer |    Model    | Hyper Parameter:Alpha | AUC  |
```

```
+------------+-------------+-----------------------+------+
|    BOW     | Naive Bayes |          1.8          | 0.62 |
|   TFIDF    | Naive Bayes |           1           | 0.64 |
+------------+-------------+-----------------------+------+
```

In [ ]: